

libstdc++

Generated by Doxygen 1.8.2

Fri Mar 1 2013 11:27:08

Contents

1	Todo List	1
2	Module Documentation	5
2.1	Extensions	5
2.1.1	Detailed Description	6
2.2	SGL	7
2.2.1	Detailed Description	9
2.2.2	Function Documentation	10
2.3	Containers	16
2.3.1	Detailed Description	16
2.4	Sequences	17
2.4.1	Detailed Description	17
2.5	Associative	19
2.5.1	Detailed Description	19
2.6	Unordered Associative	20
2.6.1	Detailed Description	20
2.7	Diagnostics	21
2.7.1	Detailed Description	21
2.8	Concurrency	22
2.8.1	Detailed Description	22
2.9	Exceptions	23
2.9.1	Detailed Description	25
2.9.2	Typedef Documentation	25
2.9.3	Function Documentation	25
2.10	Time	27
2.10.1	Detailed Description	27
2.11	Complex Numbers	28
2.11.1	Detailed Description	31
2.11.2	Function Documentation	31
2.12	Condition Variables	39
2.12.1	Detailed Description	39
2.12.2	Enumeration Type Documentation	39
2.13	Futures	40
2.13.1	Detailed Description	40
2.13.2	Enumeration Type Documentation	41
2.13.3	Function Documentation	41

2.14 I/O	42
2.14.1 Detailed Description	43
2.14.2 Typedef Documentation	43
2.15 Memory	47
2.15.1 Detailed Description	47
2.16 Pointer Abstractions	48
2.16.1 Detailed Description	50
2.16.2 Function Documentation	50
2.17 Mutexes	52
2.17.1 Detailed Description	53
2.17.2 Function Documentation	53
2.18 Numerics	55
2.18.1 Detailed Description	55
2.19 Rational Arithmetic	56
2.19.1 Detailed Description	57
2.19.2 Typedef Documentation	57
2.20 Threads	58
2.20.1 Detailed Description	58
2.21 Metaprogramming	59
2.21.1 Detailed Description	62
2.21.2 Typedef Documentation	63
2.22 Utilities	64
2.22.1 Detailed Description	67
2.22.2 Function Documentation	67
2.22.3 Variable Documentation	71
2.23 Numeric Arrays	72
2.23.1 Detailed Description	83
2.23.2 Function Documentation	83
2.24 Algorithms	103
2.24.1 Detailed Description	103
2.25 Mutating	104
2.25.1 Detailed Description	106
2.25.2 Function Documentation	106
2.26 Non-Mutating	121
2.26.1 Detailed Description	122
2.26.2 Function Documentation	122
2.27 Sorting	133

2.27.1 Detailed Description	135
2.27.2 Function Documentation	135
2.28 Set Operation	150
2.28.1 Detailed Description	150
2.28.2 Function Documentation	151
2.29 Binary Search	156
2.29.1 Detailed Description	156
2.29.2 Function Documentation	157
2.30 Allocators	161
2.30.1 Detailed Description	162
2.30.2 Typedef Documentation	162
2.31 Atomics	163
2.31.1 Detailed Description	168
2.31.2 Macro Definition Documentation	168
2.31.3 Typedef Documentation	168
2.31.4 Enumeration Type Documentation	172
2.31.5 Function Documentation	172
2.32 Hashes	173
2.32.1 Detailed Description	174
2.33 Base and Implementation Classes	175
2.33.1 Detailed Description	177
2.34 Locales	178
2.34.1 Detailed Description	179
2.34.2 Function Documentation	179
2.35 Random Number Generation	181
2.35.1 Detailed Description	181
2.35.2 Function Documentation	181
2.36 Regular Expressions	182
2.36.1 Detailed Description	187
2.36.2 Typedef Documentation	187
2.36.3 Function Documentation	188
2.37 Base and Implementation Classes	211
2.37.1 Detailed Description	212
2.37.2 Typedef Documentation	212
2.37.3 Enumeration Type Documentation	213
2.37.4 Function Documentation	213
2.37.5 Variable Documentation	213

2.38	Function Objects	214
2.38.1	Detailed Description	215
2.38.2	Function Documentation	215
2.39	Arithmetic Classes	216
2.39.1	Detailed Description	216
2.40	Comparison Classes	217
2.40.1	Detailed Description	217
2.41	Boolean Operations Classes	218
2.41.1	Detailed Description	218
2.42	Negators	219
2.42.1	Detailed Description	219
2.42.2	Function Documentation	219
2.43	Adaptors for pointers to functions	221
2.43.1	Detailed Description	221
2.43.2	Function Documentation	221
2.44	Adaptors for pointers to members	223
2.44.1	Detailed Description	224
2.45	Heap	225
2.45.1	Detailed Description	225
2.45.2	Function Documentation	225
2.46	Iterators	230
2.46.1	Detailed Description	233
2.46.2	Function Documentation	233
2.47	Iterator Tags	236
2.47.1	Detailed Description	236
2.48	Strings	237
2.48.1	Detailed Description	237
2.48.2	Typedef Documentation	237
2.49	Binder Classes	238
2.49.1	Detailed Description	239
2.49.2	Function Documentation	239
2.50	Mathematical Special Functions	241
2.50.1	Detailed Description	243
2.50.2	Function Documentation	243
2.51	Decimal Floating-Point Arithmetic	247
2.51.1	Detailed Description	247
2.52	Containers	248

2.52.1 Detailed Description	248
2.53 Hash-Based	249
2.53.1 Detailed Description	249
2.54 Base and Policy Classes	250
2.54.1 Detailed Description	250
2.55 Branch-Based	251
2.55.1 Detailed Description	251
2.56 Base and Policy Classes	252
2.56.1 Detailed Description	252
2.57 List-Based	253
2.57.1 Detailed Description	253
2.58 Exceptions	254
2.58.1 Detailed Description	254
2.59 Heap-Based	255
2.59.1 Detailed Description	256
2.59.2 Function Documentation	256
2.60 Base and Policy Classes	257
2.60.1 Detailed Description	257
2.61 Policy-Based Data Structures	258
2.61.1 Detailed Description	258
2.62 Tags	259
2.62.1 Detailed Description	259
2.62.2 Typedef Documentation	259
2.63 Invalidation Guarantees	260
2.63.1 Detailed Description	260
2.64 Data Structure Type	261
2.64.1 Detailed Description	262
2.65 Traits	263
2.65.1 Detailed Description	265
2.66 Random Number Generators	266
2.66.1 Detailed Description	267
2.66.2 Typedef Documentation	267
2.66.3 Function Documentation	268
2.67 Random Number Distributions	271
2.67.1 Detailed Description	271
2.68 Uniform Distributions	272
2.68.1 Detailed Description	273

2.68.2	Function Documentation	273
2.69	Normal Distributions	275
2.69.1	Detailed Description	276
2.69.2	Function Documentation	276
2.70	Bernoulli Distributions	278
2.70.1	Detailed Description	279
2.70.2	Function Documentation	279
2.71	Poisson Distributions	281
2.71.1	Detailed Description	282
2.71.2	Function Documentation	282
2.72	Random Number Utilities	286
2.72.1	Detailed Description	286
3	Namespace Documentation	287
3.1	__gnu_cxx Namespace Reference	287
3.1.1	Detailed Description	303
3.1.2	Function Documentation	303
3.2	__gnu_cxx::__detail Namespace Reference	312
3.2.1	Detailed Description	313
3.2.2	Function Documentation	313
3.3	__gnu_cxx::typelist Namespace Reference	313
3.3.1	Detailed Description	314
3.3.2	Function Documentation	314
3.4	__gnu_debug Namespace Reference	314
3.4.1	Detailed Description	320
3.4.2	Enumeration Type Documentation	320
3.4.3	Function Documentation	320
3.5	__gnu_internal Namespace Reference	323
3.5.1	Detailed Description	323
3.6	__gnu_parallel Namespace Reference	323
3.6.1	Detailed Description	333
3.6.2	Typedef Documentation	333
3.6.3	Enumeration Type Documentation	334
3.6.4	Function Documentation	334
3.6.5	Variable Documentation	367
3.7	__gnu_pbds Namespace Reference	368
3.7.1	Detailed Description	371

3.8	__gnu_profile Namespace Reference	371
3.8.1	Detailed Description	375
3.8.2	Typedef Documentation	375
3.8.3	Function Documentation	376
3.9	__gnu_sequential Namespace Reference	376
3.9.1	Detailed Description	376
3.10	abi Namespace Reference	376
3.10.1	Detailed Description	376
3.11	std Namespace Reference	376
3.11.1	Detailed Description	496
3.11.2	Typedef Documentation	496
3.11.3	Enumeration Type Documentation	497
3.11.4	Function Documentation	498
3.11.5	Variable Documentation	565
3.12	std::__debug Namespace Reference	565
3.12.1	Detailed Description	570
3.12.2	Function Documentation	570
3.13	std::__detail Namespace Reference	570
3.13.1	Detailed Description	574
3.14	std::__parallel Namespace Reference	574
3.14.1	Detailed Description	591
3.15	std::__profile Namespace Reference	591
3.15.1	Detailed Description	596
3.15.2	Function Documentation	596
3.16	std::chrono Namespace Reference	597
3.16.1	Detailed Description	599
3.16.2	Typedef Documentation	599
3.16.3	Function Documentation	600
3.17	std::decimal Namespace Reference	600
3.17.1	Detailed Description	609
3.17.2	Function Documentation	609
3.18	std::placeholders Namespace Reference	609
3.18.1	Detailed Description	610
3.19	std::regex_constants Namespace Reference	610
3.19.1	Detailed Description	611
3.19.2	Typedef Documentation	612
3.19.3	Enumeration Type Documentation	612

3.19.4	Function Documentation	612
3.19.5	Variable Documentation	613
3.20	std::rel_ops Namespace Reference	616
3.20.1	Detailed Description	617
3.20.2	Function Documentation	617
3.21	std::this_thread Namespace Reference	618
3.21.1	Detailed Description	618
3.21.2	Function Documentation	618
3.22	std::tr1 Namespace Reference	619
3.22.1	Detailed Description	622
3.23	std::tr1::__detail Namespace Reference	622
3.23.1	Detailed Description	622
3.24	std::tr2 Namespace Reference	622
3.24.1	Detailed Description	624
3.24.2	Function Documentation	624
3.25	std::tr2::__detail Namespace Reference	627
3.25.1	Detailed Description	627
4	Class Documentation	627
4.1	__cxxabiv1::__forced_unwind Class Reference	627
4.1.1	Detailed Description	627
4.2	__gnu_cxx::__alloc_traits<_Alloc> Struct Template Reference	628
4.2.1	Detailed Description	629
4.2.2	Member Typedef Documentation	629
4.2.3	Member Function Documentation	630
4.3	__gnu_cxx::__common_pool_policy<_PoolTp, _Thread> Struct Template Reference	632
4.3.1	Detailed Description	632
4.4	__gnu_cxx::__detail::__mini_vector<_Tp> Class Template Reference	633
4.4.1	Detailed Description	633
4.5	__gnu_cxx::__detail::__Bitmap_counter<_Tp> Class Template Reference	634
4.5.1	Detailed Description	634
4.6	__gnu_cxx::__detail::__Ffit_finder<_Tp> Class Template Reference	635
4.6.1	Detailed Description	635
4.6.2	Member Typedef Documentation	635
4.7	__gnu_cxx::__mt_alloc<_Tp, _Poolp> Class Template Reference	636
4.7.1	Detailed Description	637
4.8	__gnu_cxx::__mt_alloc_base<_Tp> Class Template Reference	637

4.8.1 Detailed Description	638
4.9 __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct Template Reference	638
4.9.1 Detailed Description	638
4.10 __gnu_cxx::__pool< false > Class Template Reference	639
4.10.1 Detailed Description	640
4.11 __gnu_cxx::__pool< true > Class Template Reference	640
4.11.1 Detailed Description	641
4.12 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference	641
4.12.1 Detailed Description	642
4.13 __gnu_cxx::__pool_alloc_base Class Reference	643
4.13.1 Detailed Description	643
4.14 __gnu_cxx::__pool_base Struct Reference	644
4.14.1 Detailed Description	644
4.15 __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference	645
4.15.1 Detailed Description	646
4.16 __gnu_cxx::__scoped_lock Class Reference	647
4.16.1 Detailed Description	647
4.17 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference	647
4.17.1 Detailed Description	651
4.17.2 Constructor & Destructor Documentation	651
4.17.3 Member Function Documentation	654
4.17.4 Member Data Documentation	695
4.18 __gnu_cxx::__Caster< _ToType > Struct Template Reference	695
4.18.1 Detailed Description	695
4.19 __gnu_cxx::__Char_types< _CharT > Struct Template Reference	696
4.19.1 Detailed Description	696
4.20 __gnu_cxx::__ExtPtr_allocator< _Tp > Class Template Reference	696
4.20.1 Detailed Description	697
4.21 __gnu_cxx::__Invalid_type Struct Reference	697
4.21.1 Detailed Description	697
4.22 __gnu_cxx::__Pointer_adapter< _Storage_policy > Class Template Reference	698
4.22.1 Detailed Description	699
4.23 __gnu_cxx::__Relative_pointer_impl< _Tp > Class Template Reference	700
4.23.1 Detailed Description	700
4.24 __gnu_cxx::__Relative_pointer_impl< const _Tp > Class Template Reference	700
4.24.1 Detailed Description	701
4.25 __gnu_cxx::__Std_pointer_impl< _Tp > Class Template Reference	701

4.25.1 Detailed Description	701
4.26 __gnu_cxx::Unqualified_type< _Tp > Struct Template Reference	701
4.26.1 Detailed Description	702
4.27 __gnu_cxx::annotate_base Struct Reference	702
4.27.1 Detailed Description	702
4.28 __gnu_cxx::array_allocator< _Tp, _Array > Class Template Reference	703
4.28.1 Detailed Description	704
4.29 __gnu_cxx::array_allocator_base< _Tp > Class Template Reference	704
4.29.1 Detailed Description	705
4.30 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference . .	705
4.30.1 Detailed Description	706
4.30.2 Member Typedef Documentation	706
4.31 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference	706
4.31.1 Detailed Description	707
4.31.2 Member Function Documentation	708
4.32 __gnu_cxx::char_traits< _CharT > Struct Template Reference	709
4.32.1 Detailed Description	710
4.33 __gnu_cxx::character< V, I, S > Struct Template Reference	710
4.33.1 Detailed Description	710
4.34 __gnu_cxx::condition_base Struct Reference	711
4.34.1 Detailed Description	711
4.35 __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference	711
4.35.1 Detailed Description	712
4.36 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference	712
4.36.1 Detailed Description	712
4.37 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference	712
4.37.1 Detailed Description	713
4.38 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference	713
4.38.1 Detailed Description	713
4.39 __gnu_cxx::enc_filebuf< _CharT > Class Template Reference	714
4.39.1 Detailed Description	716
4.39.2 Member Function Documentation	716
4.39.3 Member Data Documentation	728
4.40 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference	731
4.40.1 Detailed Description	732
4.41 __gnu_cxx::encoding_state Class Reference	732
4.41.1 Detailed Description	733

4.42	__gnu_cxx::forced_error Struct Reference	733
4.42.1	Detailed Description	734
4.42.2	Member Function Documentation	734
4.43	__gnu_cxx::free_list Class Reference	734
4.43.1	Detailed Description	735
4.43.2	Member Function Documentation	735
4.44	__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc> Class Template Reference	735
4.44.1	Detailed Description	737
4.45	__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc> Class Template Reference	737
4.45.1	Detailed Description	739
4.46	__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc> Class Template Reference	739
4.46.1	Detailed Description	740
4.47	__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc> Class Template Reference	741
4.47.1	Detailed Description	742
4.48	__gnu_cxx::limit_condition Struct Reference	742
4.48.1	Detailed Description	743
4.49	__gnu_cxx::limit_condition::always_adjustor Struct Reference	743
4.49.1	Detailed Description	743
4.50	__gnu_cxx::limit_condition::limit_adjustor Struct Reference	743
4.50.1	Detailed Description	743
4.51	__gnu_cxx::limit_condition::never_adjustor Struct Reference	743
4.51.1	Detailed Description	744
4.52	__gnu_cxx::malloc_allocator<_Tp> Class Template Reference	744
4.52.1	Detailed Description	744
4.53	__gnu_cxx::new_allocator<_Tp> Class Template Reference	745
4.53.1	Detailed Description	746
4.54	__gnu_cxx::project1st<_Arg1, _Arg2> Struct Template Reference	746
4.54.1	Detailed Description	746
4.54.2	Member Typedef Documentation	746
4.55	__gnu_cxx::project2nd<_Arg1, _Arg2> Struct Template Reference	747
4.55.1	Detailed Description	747
4.55.2	Member Typedef Documentation	747
4.56	__gnu_cxx::random_condition Struct Reference	748
4.56.1	Detailed Description	748
4.57	__gnu_cxx::random_condition::always_adjustor Struct Reference	749
4.57.1	Detailed Description	749
4.58	__gnu_cxx::random_condition::group_adjustor Struct Reference	749

4.58.1 Detailed Description	749
4.59 __gnu_cxx::random_condition::never_adjustor Struct Reference	749
4.59.1 Detailed Description	749
4.60 __gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference	749
4.60.1 Detailed Description	752
4.61 __gnu_cxx::recursive_init_error Class Reference	753
4.61.1 Detailed Description	753
4.61.2 Member Function Documentation	753
4.62 __gnu_cxx::rope<_CharT, _Alloc > Class Template Reference	753
4.62.1 Detailed Description	759
4.63 __gnu_cxx::select1st<_Pair > Struct Template Reference	759
4.63.1 Detailed Description	759
4.63.2 Member Typedef Documentation	759
4.64 __gnu_cxx::select2nd<_Pair > Struct Template Reference	760
4.64.1 Detailed Description	760
4.64.2 Member Typedef Documentation	760
4.65 __gnu_cxx::slist<_Tp, _Alloc > Class Template Reference	760
4.65.1 Detailed Description	763
4.66 __gnu_cxx::stdio_filebuf<_CharT, _Traits > Class Template Reference	763
4.66.1 Detailed Description	766
4.66.2 Constructor & Destructor Documentation	766
4.66.3 Member Function Documentation	767
4.66.4 Member Data Documentation	781
4.67 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits > Class Template Reference	784
4.67.1 Detailed Description	787
4.67.2 Member Typedef Documentation	787
4.67.3 Member Function Documentation	787
4.67.4 Member Data Documentation	799
4.68 __gnu_cxx::subtractive_rng Class Reference	800
4.68.1 Detailed Description	801
4.68.2 Member Typedef Documentation	801
4.68.3 Constructor & Destructor Documentation	801
4.68.4 Member Function Documentation	801
4.69 __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp > Struct Template Reference	802
4.69.1 Detailed Description	803
4.69.2 Constructor & Destructor Documentation	803
4.69.3 Member Function Documentation	803

4.70	__gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference	804
4.70.1	Detailed Description	805
4.71	__gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference	805
4.71.1	Detailed Description	806
4.72	__gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference	806
4.72.1	Detailed Description	807
4.73	__gnu_cxx::throw_value_base< _Cond > Struct Template Reference	808
4.73.1	Detailed Description	808
4.74	__gnu_cxx::throw_value_limit Struct Reference	809
4.74.1	Detailed Description	810
4.75	__gnu_cxx::throw_value_random Struct Reference	810
4.75.1	Detailed Description	811
4.76	__gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference	811
4.76.1	Detailed Description	812
4.76.2	Member Typedef Documentation	812
4.77	__gnu_debug::_After_nth_from< _Iterator > Class Template Reference	812
4.77.1	Detailed Description	813
4.78	__gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference	813
4.78.1	Detailed Description	813
4.79	__gnu_debug::_Equal_to< _Type > Class Template Reference	813
4.79.1	Detailed Description	813
4.80	__gnu_debug::_Not_equal_to< _Type > Class Template Reference	814
4.80.1	Detailed Description	814
4.81	__gnu_debug::_Safe_iterator< _Iterator, _Sequence > Class Template Reference	814
4.81.1	Detailed Description	816
4.81.2	Constructor & Destructor Documentation	816
4.81.3	Member Function Documentation	817
4.81.4	Member Data Documentation	822
4.82	__gnu_debug::_Safe_iterator_base Class Reference	823
4.82.1	Detailed Description	824
4.82.2	Constructor & Destructor Documentation	824
4.82.3	Member Function Documentation	825
4.82.4	Member Data Documentation	826
4.83	__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence > Class Template Reference	827
4.83.1	Detailed Description	828
4.83.2	Constructor & Destructor Documentation	829
4.83.3	Member Function Documentation	829

4.83.4	Member Data Documentation	833
4.84	__gnu_debug::__Safe_local_iterator_base Class Reference	835
4.84.1	Detailed Description	836
4.84.2	Constructor & Destructor Documentation	836
4.84.3	Member Function Documentation	836
4.84.4	Member Data Documentation	838
4.85	__gnu_debug::__Safe_sequence< _Sequence > Class Template Reference	839
4.85.1	Detailed Description	840
4.85.2	Member Function Documentation	840
4.85.3	Member Data Documentation	841
4.86	__gnu_debug::__Safe_sequence_base Class Reference	842
4.86.1	Detailed Description	843
4.86.2	Constructor & Destructor Documentation	843
4.86.3	Member Function Documentation	843
4.86.4	Member Data Documentation	844
4.87	__gnu_debug::__Safe_unordered_container< _Container > Class Template Reference	845
4.87.1	Detailed Description	846
4.87.2	Member Function Documentation	846
4.87.3	Member Data Documentation	848
4.88	__gnu_debug::__Safe_unordered_container_base Class Reference	849
4.88.1	Detailed Description	850
4.88.2	Constructor & Destructor Documentation	850
4.88.3	Member Function Documentation	850
4.88.4	Member Data Documentation	851
4.89	__gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference	852
4.89.1	Detailed Description	857
4.89.2	Member Function Documentation	857
4.89.3	Member Data Documentation	873
4.90	__gnu_parallel::__accumulate_binop_reduct< _BinOp > Struct Template Reference	873
4.90.1	Detailed Description	874
4.91	__gnu_parallel::__accumulate_selector< _It > Struct Template Reference	874
4.91.1	Detailed Description	874
4.91.2	Member Function Documentation	875
4.91.3	Member Data Documentation	875
4.92	__gnu_parallel::__adjacent_difference_selector< _It > Struct Template Reference	875
4.92.1	Detailed Description	876
4.92.2	Member Data Documentation	876

4.93	__gnu_parallel::__adjacent_find_selector Struct Reference	876
4.93.1	Detailed Description	877
4.93.2	Member Function Documentation	877
4.94	__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference	878
4.94.1	Detailed Description	878
4.94.2	Member Typedef Documentation	879
4.95	__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference	880
4.95.1	Detailed Description	880
4.95.2	Member Typedef Documentation	881
4.96	__gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference	881
4.96.1	Detailed Description	882
4.96.2	Member Function Documentation	882
4.96.3	Member Data Documentation	882
4.97	__gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference	883
4.97.1	Detailed Description	883
4.97.2	Member Function Documentation	883
4.97.3	Member Data Documentation	884
4.98	__gnu_parallel::__fill_selector< _It > Struct Template Reference	884
4.98.1	Detailed Description	884
4.98.2	Member Function Documentation	885
4.98.3	Member Data Documentation	885
4.99	__gnu_parallel::__find_first_of_selector< _FIterator > Struct Template Reference	885
4.99.1	Detailed Description	886
4.99.2	Member Function Documentation	886
4.100	__gnu_parallel::__find_if_selector Struct Reference	887
4.100.1	Detailed Description	887
4.100.2	Member Function Documentation	887
4.101	__gnu_parallel::__for_each_selector< _It > Struct Template Reference	888
4.101.1	Detailed Description	889
4.101.2	Member Function Documentation	889
4.101.3	Member Data Documentation	889
4.102	__gnu_parallel::__generate_selector< _It > Struct Template Reference	890
4.102.1	Detailed Description	890
4.102.2	Member Function Documentation	890
4.102.3	Member Data Documentation	891

4.103 __gnu_parallel::__generic_find_selector Struct Reference	891
4.103.1 Detailed Description	891
4.104 __gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference	892
4.104.1 Detailed Description	893
4.104.2 Member Data Documentation	893
4.105 __gnu_parallel::__identity_selector< _It > Struct Template Reference	893
4.105.1 Detailed Description	894
4.105.2 Member Function Documentation	894
4.105.3 Member Data Documentation	894
4.106 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference	895
4.106.1 Detailed Description	895
4.106.2 Constructor & Destructor Documentation	895
4.106.3 Member Function Documentation	896
4.106.4 Member Data Documentation	896
4.107 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference	896
4.107.1 Detailed Description	897
4.108 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference	897
4.108.1 Detailed Description	897
4.109 __gnu_parallel::__mismatch_selector Struct Reference	898
4.109.1 Detailed Description	898
4.109.2 Member Function Documentation	898
4.110 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	899
4.110.1 Detailed Description	899
4.111 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	899
4.111.1 Detailed Description	899
4.112 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	900
4.112.1 Detailed Description	900
4.113 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	900
4.113.1 Detailed Description	900
4.114 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	901
4.114.1 Detailed Description	901
4.115 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	901
4.115.1 Detailed Description	901

4.116	__gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Template Reference	902
4.116.1	Detailed Description	902
4.116.2	Constructor & Destructor Documentation	902
4.116.3	Member Function Documentation	903
4.116.4	Member Data Documentation	903
4.117	__gnu_parallel::__replace_selector< _It, _Tp > Struct Template Reference	904
4.117.1	Detailed Description	904
4.117.2	Constructor & Destructor Documentation	904
4.117.3	Member Function Documentation	905
4.117.4	Member Data Documentation	905
4.118	__gnu_parallel::__transform1_selector< _It > Struct Template Reference	905
4.118.1	Detailed Description	906
4.118.2	Member Function Documentation	906
4.118.3	Member Data Documentation	906
4.119	__gnu_parallel::__transform2_selector< _It > Struct Template Reference	907
4.119.1	Detailed Description	907
4.119.2	Member Function Documentation	907
4.119.3	Member Data Documentation	908
4.120	__gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference	908
4.120.1	Detailed Description	909
4.120.2	Member Typedef Documentation	909
4.121	__gnu_parallel::__DRandomShufflingGlobalData< _RAIter > Struct Template Reference	909
4.121.1	Detailed Description	910
4.121.2	Constructor & Destructor Documentation	910
4.121.3	Member Data Documentation	910
4.122	__gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference	911
4.122.1	Detailed Description	911
4.122.2	Member Data Documentation	911
4.123	__gnu_parallel::__DummyReduct Struct Reference	912
4.123.1	Detailed Description	912
4.124	__gnu_parallel::__EqualFromLess< _T1, _T2, _Compare > Class Template Reference	913
4.124.1	Detailed Description	913
4.124.2	Member Typedef Documentation	913
4.125	__gnu_parallel::__EqualTo< _T1, _T2 > Struct Template Reference	914
4.125.1	Detailed Description	915
4.125.2	Member Typedef Documentation	915
4.126	__gnu_parallel::__GuardedIterator< _RAIter, _Compare > Class Template Reference	915

4.126.1 Detailed Description	916
4.126.2 Constructor & Destructor Documentation	916
4.126.3 Member Function Documentation	916
4.126.4 Friends And Related Function Documentation	917
4.127 __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference	918
4.127.1 Detailed Description	919
4.127.2 Member Typedef Documentation	919
4.127.3 Member Data Documentation	919
4.128 __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference	919
4.128.1 Detailed Description	920
4.129 __gnu_parallel::_Job< _DifferenceTp > Struct Template Reference	920
4.129.1 Detailed Description	921
4.129.2 Member Data Documentation	921
4.130 __gnu_parallel::_Less< _T1, _T2 > Struct Template Reference	922
4.130.1 Detailed Description	922
4.130.2 Member Typedef Documentation	922
4.131 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference	923
4.131.1 Detailed Description	924
4.131.2 Member Typedef Documentation	924
4.132 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference	925
4.132.1 Detailed Description	925
4.132.2 Member Typedef Documentation	925
4.133 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference	926
4.133.1 Detailed Description	927
4.133.2 Member Function Documentation	927
4.133.3 Member Data Documentation	927
4.134 __gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference	929
4.134.1 Detailed Description	929
4.134.2 Member Function Documentation	930
4.134.3 Member Data Documentation	931
4.135 __gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference	932
4.135.1 Detailed Description	932
4.135.2 Constructor & Destructor Documentation	933
4.135.3 Member Function Documentation	933
4.135.4 Member Data Documentation	934
4.136 __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference	934

4.136.1 Detailed Description	935
4.136.2 Member Data Documentation	935
4.137 <code>__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare ></code> Class Template Reference	936
4.137.1 Detailed Description	936
4.138 <code>__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare ></code> Class Template Reference	937
4.138.1 Detailed Description	937
4.139 <code>__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare ></code> Class Template Reference	938
4.139.1 Detailed Description	938
4.140 <code>__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser</code> Struct Reference	939
4.140.1 Detailed Description	939
4.141 <code>__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare ></code> Class Template Reference	939
4.141.1 Detailed Description	940
4.142 <code>__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare ></code> Class Template Reference	940
4.142.1 Detailed Description	941
4.143 <code>__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare ></code> Class Template Reference	941
4.143.1 Detailed Description	942
4.144 <code>__gnu_parallel::_LoserTreeTraits< _Tp ></code> Struct Template Reference	942
4.144.1 Detailed Description	942
4.144.2 Member Data Documentation	942
4.145 <code>__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare ></code> Class Template Reference	943
4.145.1 Detailed Description	944
4.146 <code>__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare ></code> Class Template Reference	944
4.146.1 Detailed Description	945
4.147 <code>__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare ></code> Class Template Reference	945
4.147.1 Detailed Description	945
4.148 <code>__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result ></code> Struct Template Reference	946
4.148.1 Detailed Description	946
4.148.2 Member Typedef Documentation	947
4.149 <code>__gnu_parallel::_Nothing</code> Struct Reference	947
4.149.1 Detailed Description	947
4.149.2 Member Function Documentation	947
4.150 <code>__gnu_parallel::_Piece< _DifferenceTp ></code> Struct Template Reference	948
4.150.1 Detailed Description	948
4.150.2 Member Data Documentation	948
4.151 <code>__gnu_parallel::_Plus< _Tp1, _Tp2, _Result ></code> Struct Template Reference	949
4.151.1 Detailed Description	949
4.151.2 Member Typedef Documentation	949

4.152	__gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference	950
4.152.1	Detailed Description	950
4.152.2	Member Data Documentation	950
4.153	__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference	951
4.153.1	Detailed Description	952
4.153.2	Constructor & Destructor Documentation	952
4.153.3	Member Function Documentation	952
4.154	__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference	953
4.154.1	Detailed Description	953
4.155	__gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference	953
4.155.1	Detailed Description	954
4.155.2	Member Typedef Documentation	954
4.155.3	Constructor & Destructor Documentation	954
4.155.4	Member Data Documentation	954
4.156	__gnu_parallel::_RandomNumber Class Reference	955
4.156.1	Detailed Description	955
4.156.2	Constructor & Destructor Documentation	956
4.156.3	Member Function Documentation	956
4.157	__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference	957
4.157.1	Detailed Description	957
4.157.2	Constructor & Destructor Documentation	957
4.157.3	Member Function Documentation	958
4.158	__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference	958
4.158.1	Detailed Description	958
4.159	__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference	959
4.159.1	Detailed Description	959
4.160	__gnu_parallel::_Settings Struct Reference	959
4.160.1	Detailed Description	960
4.160.2	Member Function Documentation	961
4.160.3	Member Data Documentation	961
4.161	__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference	966
4.161.1	Detailed Description	966
4.162	__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference	966
4.162.1	Detailed Description	966
4.163	__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference	966

4.163.1 Detailed Description	966
4.164 __gnu_parallel::balanced_quicksort_tag Struct Reference	967
4.164.1 Detailed Description	967
4.164.2 Member Function Documentation	967
4.165 __gnu_parallel::balanced_tag Struct Reference	968
4.165.1 Detailed Description	968
4.165.2 Member Function Documentation	968
4.166 __gnu_parallel::constant_size_blocks_tag Struct Reference	969
4.166.1 Detailed Description	969
4.167 __gnu_parallel::default_parallel_tag Struct Reference	970
4.167.1 Detailed Description	970
4.167.2 Member Function Documentation	970
4.168 __gnu_parallel::equal_split_tag Struct Reference	971
4.168.1 Detailed Description	971
4.169 __gnu_parallel::exact_tag Struct Reference	972
4.169.1 Detailed Description	972
4.169.2 Member Function Documentation	972
4.170 __gnu_parallel::find_tag Struct Reference	973
4.170.1 Detailed Description	973
4.171 __gnu_parallel::growing_blocks_tag Struct Reference	974
4.171.1 Detailed Description	974
4.172 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	974
4.172.1 Detailed Description	975
4.172.2 Member Function Documentation	975
4.173 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	976
4.173.1 Detailed Description	976
4.173.2 Member Function Documentation	976
4.174 __gnu_parallel::multiway_mergesort_tag Struct Reference	977
4.174.1 Detailed Description	977
4.174.2 Member Function Documentation	977
4.175 __gnu_parallel::omp_loop_static_tag Struct Reference	978
4.175.1 Detailed Description	978
4.175.2 Member Function Documentation	978
4.176 __gnu_parallel::omp_loop_tag Struct Reference	979
4.176.1 Detailed Description	979
4.176.2 Member Function Documentation	980
4.177 __gnu_parallel::parallel_tag Struct Reference	981

4.177.1 Detailed Description	982
4.177.2 Constructor & Destructor Documentation	982
4.177.3 Member Function Documentation	982
4.178 <code>_gnu_parallel::quicksort_tag</code> Struct Reference	983
4.178.1 Detailed Description	983
4.178.2 Member Function Documentation	983
4.179 <code>_gnu_parallel::sampling_tag</code> Struct Reference	984
4.179.1 Detailed Description	984
4.179.2 Member Function Documentation	984
4.180 <code>_gnu_parallel::sequential_tag</code> Struct Reference	985
4.180.1 Detailed Description	985
4.181 <code>_gnu_parallel::unbalanced_tag</code> Struct Reference	985
4.181.1 Detailed Description	985
4.181.2 Member Function Documentation	986
4.182 <code>_gnu_pbds::associative_tag</code> Struct Reference	986
4.182.1 Detailed Description	986
4.183 <code>_gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc ></code> Class Template Reference	987
4.183.1 Detailed Description	987
4.184 <code>_gnu_pbds::basic_branch_tag</code> Struct Reference	988
4.184.1 Detailed Description	988
4.185 <code>_gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc ></code> Class Template Reference	989
4.185.1 Detailed Description	989
4.186 <code>_gnu_pbds::basic_hash_tag</code> Struct Reference	990
4.186.1 Detailed Description	990
4.187 <code>_gnu_pbds::basic_invalidation_guarantee</code> Struct Reference	991
4.187.1 Detailed Description	991
4.188 <code>_gnu_pbds::binary_heap_tag</code> Struct Reference	992
4.188.1 Detailed Description	992
4.189 <code>_gnu_pbds::binomial_heap_tag</code> Struct Reference	993
4.189.1 Detailed Description	993
4.190 <code>_gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type ></code> Class Template Reference	993
4.190.1 Detailed Description	994
4.190.2 Member Enumeration Documentation	994
4.190.3 Constructor & Destructor Documentation	994
4.190.4 Member Function Documentation	995

4.191 __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store-Hash, _Alloc > Class Template Reference	997
4.191.1 Detailed Description	998
4.191.2 Constructor & Destructor Documentation	999
4.192 __gnu_pbds::cc_hash_tag Struct Reference	1002
4.192.1 Detailed Description	1002
4.193 __gnu_pbds::container_error Struct Reference	1003
4.193.1 Detailed Description	1003
4.193.2 Member Function Documentation	1003
4.194 __gnu_pbds::container_tag Struct Reference	1004
4.194.1 Detailed Description	1004
4.195 __gnu_pbds::container_traits< Cntnr > Struct Template Reference	1005
4.195.1 Detailed Description	1005
4.195.2 Member Enumeration Documentation	1005
4.196 __gnu_pbds::container_traits_base< binary_heap_tag > Struct Template Reference	1006
4.196.1 Detailed Description	1006
4.197 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Template Reference	1006
4.197.1 Detailed Description	1006
4.198 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Template Reference	1007
4.198.1 Detailed Description	1007
4.199 __gnu_pbds::container_traits_base< gp_hash_tag > Struct Template Reference	1007
4.199.1 Detailed Description	1007
4.200 __gnu_pbds::container_traits_base< list_update_tag > Struct Template Reference	1007
4.200.1 Detailed Description	1008
4.201 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Template Reference	1008
4.201.1 Detailed Description	1008
4.202 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Template Reference	1008
4.202.1 Detailed Description	1008
4.203 __gnu_pbds::container_traits_base< pat_trie_tag > Struct Template Reference	1009
4.203.1 Detailed Description	1009
4.204 __gnu_pbds::container_traits_base< rb_tree_tag > Struct Template Reference	1009
4.204.1 Detailed Description	1009
4.205 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Template Reference	1009
4.205.1 Detailed Description	1010
4.206 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Template Reference	1010
4.206.1 Detailed Description	1010
4.207 __gnu_pbds::container_traits_base< thin_heap_tag > Struct Template Reference	1010

4.207.1 Detailed Description	1010
4.208 <code>_gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference</code>	1011
4.208.1 Detailed Description	1012
4.209 <code>_gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference</code>	1013
4.209.1 Detailed Description	1014
4.209.2 Member Typedef Documentation	1014
4.209.3 Member Function Documentation	1015
4.210 <code>_gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference</code>	1016
4.210.1 Detailed Description	1018
4.211 <code>_gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference</code>	1018
4.211.1 Detailed Description	1019
4.211.2 Member Typedef Documentation	1019
4.211.3 Member Function Documentation	1020
4.212 <code>_gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference</code>	1022
4.212.1 Detailed Description	1023
4.212.2 Member Typedef Documentation	1024
4.213 <code>_gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference</code>	1024
4.213.1 Detailed Description	1025
4.213.2 Member Typedef Documentation	1025
4.214 <code>_gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference</code>	1026
4.214.1 Detailed Description	1028
4.215 <code>_gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference</code>	1028
4.215.1 Detailed Description	1029
4.215.2 Member Typedef Documentation	1029
4.215.3 Constructor & Destructor Documentation	1030
4.215.4 Member Function Documentation	1030
4.216 <code>_gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference</code>	1032
4.216.1 Detailed Description	1033
4.216.2 Member Typedef Documentation	1033
4.216.3 Constructor & Destructor Documentation	1034
4.216.4 Member Function Documentation	1034

4.217__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	1035
4.217.1 Detailed Description	1037
4.218__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference .	1037
4.218.1 Detailed Description	1039
4.219__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc > Struct Template Reference	1039
4.219.1 Detailed Description	1040
4.220__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc > Struct Template Reference	1040
4.220.1 Detailed Description	1041
4.221__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > Class Template Reference	1041
4.221.1 Detailed Description	1043
4.221.2 Member Enumeration Documentation	1044
4.221.3 Member Function Documentation	1044
4.222__gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference	1046
4.222.1 Detailed Description	1046
4.223__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference	1047
4.223.1 Detailed Description	1047
4.223.2 Member Typedef Documentation	1047
4.224__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference	1047
4.224.1 Detailed Description	1047
4.224.2 Member Typedef Documentation	1047
4.225__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference	1048
4.225.1 Detailed Description	1048
4.225.2 Member Typedef Documentation	1048
4.226__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference	1048
4.226.1 Detailed Description	1048
4.226.2 Member Typedef Documentation	1049
4.227__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference	1049
4.227.1 Detailed Description	1049
4.227.2 Member Typedef Documentation	1049
4.228__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference	1049
4.228.1 Detailed Description	1050
4.228.2 Member Typedef Documentation	1050

4.229 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI > Struct</code> Template Reference	1050
4.229.1 Detailed Description	1050
4.229.2 Member Typedef Documentation	1050
4.230 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI ></code> Struct Template Reference	1051
4.230.1 Detailed Description	1051
4.230.2 Member Typedef Documentation	1051
4.231 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI > Struct</code> Template Reference	1051
4.231.1 Detailed Description	1051
4.231.2 Member Typedef Documentation	1051
4.232 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI > Struct</code> Template Reference	1052
4.232.1 Detailed Description	1052
4.233 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI > Struct</code> Template Reference	1052
4.233.1 Detailed Description	1052
4.233.2 Member Typedef Documentation	1052
4.234 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI > Struct</code> Template Reference	1053
4.234.1 Detailed Description	1053
4.234.2 Member Typedef Documentation	1053
4.235 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI > Struct</code> Template Reference	1053
4.235.1 Detailed Description	1054
4.235.2 Member Typedef Documentation	1054
4.236 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI > Struct</code> Template Reference	1054
4.236.1 Detailed Description	1054
4.236.2 Member Typedef Documentation	1054
4.237 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI ></code> Struct Template Reference	1055
4.237.1 Detailed Description	1055
4.237.2 Member Typedef Documentation	1055
4.238 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI > Struct</code> Template Reference	1055
4.238.1 Detailed Description	1055
4.238.2 Member Typedef Documentation	1055

4.239	__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI > Struct Template Reference	1056
4.239.1	Detailed Description	1056
4.239.2	Member Typedef Documentation	1056
4.240	__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI > Struct Template Reference	1056
4.240.1	Detailed Description	1056
4.241	__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI > Struct Template Reference	1057
4.241.1	Detailed Description	1057
4.241.2	Member Typedef Documentation	1057
4.242	__gnu_pbds::detail::default_comb_hash_fn Struct Reference	1057
4.242.1	Detailed Description	1057
4.242.2	Member Typedef Documentation	1057
4.243	__gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference	1058
4.243.1	Detailed Description	1058
4.243.2	Member Typedef Documentation	1058
4.244	__gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference	1058
4.244.1	Detailed Description	1058
4.244.2	Member Typedef Documentation	1058
4.245	__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference	1059
4.245.1	Detailed Description	1059
4.245.2	Member Typedef Documentation	1059
4.246	__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference	1059
4.246.1	Detailed Description	1059
4.246.2	Member Typedef Documentation	1060
4.247	__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference	1060
4.247.1	Detailed Description	1060
4.247.2	Member Typedef Documentation	1060
4.248	__gnu_pbds::detail::default_update_policy Struct Reference	1060
4.248.1	Detailed Description	1061
4.248.2	Member Typedef Documentation	1061
4.249	__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference	1061
4.249.1	Detailed Description	1061
4.250	__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false > Struct Template Reference	1061
4.250.1	Detailed Description	1062
4.251	__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference	1062

4.251.1 Detailed Description	1062
4.252 <code>__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true ></code> Struct Template Reference	1062
4.252.1 Detailed Description	1062
4.252.2 Member Typedef Documentation	1063
4.253 <code>__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false ></code> Struct Template Reference	1063
4.253.1 Detailed Description	1063
4.254 <code>__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true ></code> Struct Template Reference	1063
4.254.1 Detailed Description	1063
4.255 <code>__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn ></code> Struct Template Reference	1063
4.255.1 Detailed Description	1064
4.256 <code>__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy ></code> Class Template Reference	1064
4.256.1 Detailed Description	1066
4.256.2 Member Enumeration Documentation	1067
4.256.3 Member Function Documentation	1067
4.257 <code>__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false ></code> Struct Template Reference	1069
4.257.1 Detailed Description	1069
4.258 <code>__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true ></code> Struct Template Reference	1069
4.258.1 Detailed Description	1070
4.259 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true ></code> Class Template Reference	1070
4.259.1 Detailed Description	1070
4.260 <code>__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc ></code> Class Template Reference	1071
4.260.1 Detailed Description	1072
4.261 <code>__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc ></code> Class Template Reference	1073
4.261.1 Detailed Description	1074
4.261.2 Member Typedef Documentation	1074
4.261.3 Constructor & Destructor Documentation	1075
4.261.4 Member Function Documentation	1075
4.262 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node< _Value, _Metadata, _Alloc ></code> Struct Template Reference	1076
4.262.1 Detailed Description	1076
4.263 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc ></code> Class Template Reference	1077
4.263.1 Detailed Description	1078
4.263.2 Member Typedef Documentation	1078
4.263.3 Constructor & Destructor Documentation	1079

4.263.4 Member Function Documentation	1079
4.264 <code>__gnu_pbds::detail::lu_counter_metadata< Size_Type ></code> Class Template Reference	1080
4.264.1 Detailed Description	1080
4.265 <code>__gnu_pbds::detail::lu_counter_policy_base< Size_Type ></code> Class Template Reference	1081
4.265.1 Detailed Description	1081
4.266 <code>__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy ></code> Class Template Reference	1082
4.266.1 Detailed Description	1083
4.267 <code>__gnu_pbds::detail::mask_based_range_hashing< Size_Type ></code> Class Template Reference	1084
4.267.1 Detailed Description	1084
4.268 <code>__gnu_pbds::detail::mod_based_range_hashing< Size_Type ></code> Class Template Reference	1085
4.268.1 Detailed Description	1085
4.269 <code>__gnu_pbds::detail::no_throw_copies< Key, Mapped ></code> Struct Template Reference	1085
4.269.1 Detailed Description	1086
4.270 <code>__gnu_pbds::detail::no_throw_copies< Key, null_type ></code> Struct Template Reference	1086
4.270.1 Detailed Description	1086
4.271 <code>__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc ></code> Class Template Reference	1086
4.271.1 Detailed Description	1088
4.271.2 Member Function Documentation	1088
4.272 <code>__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type ></code> Class Template Reference	1089
4.272.1 Detailed Description	1090
4.273 <code>__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc ></code> Class Template Reference	1090
4.273.1 Detailed Description	1092
4.273.2 Member Function Documentation	1092
4.274 <code>__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc ></code> Class Template Reference	1092
4.274.1 Detailed Description	1093
4.274.2 Member Function Documentation	1094
4.275 <code>__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc ></code> Class Template Reference	1094
4.275.1 Detailed Description	1096
4.276 <code>__gnu_pbds::detail::pat_trie_base</code> Struct Reference	1097
4.276.1 Detailed Description	1098
4.276.2 Member Enumeration Documentation	1098
4.277 <code>__gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator ></code> Class Template Reference	1098
4.277.1 Detailed Description	1100
4.278 <code>__gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata ></code> Struct Template Reference	1100

4.278.1 Detailed Description	1101
4.279 <code>__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata></code> Struct Template Reference	1101
4.279.1 Detailed Description	1103
4.280 <code>__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator</code> Struct Reference	1103
4.280.1 Detailed Description	1104
4.281 <code>__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator</code> Struct Reference	1104
4.281.1 Detailed Description	1105
4.282 <code>__gnu_pbds::detail::pat_trie_base::_Iter<Node, Leaf, Head, Inode, Is_Forward_Iterator></code> Class Template Reference	1105
4.282.1 Detailed Description	1107
4.283 <code>__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata></code> Struct Template Reference	1107
4.283.1 Detailed Description	1108
4.284 <code>__gnu_pbds::detail::pat_trie_base::_Metadata<Metadata, _Alloc></code> Struct Template Reference	1108
4.284.1 Detailed Description	1109
4.285 <code>__gnu_pbds::detail::pat_trie_base::_Metadata<null_type, _Alloc></code> Struct Template Reference	1109
4.285.1 Detailed Description	1109
4.286 <code>__gnu_pbds::detail::pat_trie_base::_Node_base<_ATraits, Metadata></code> Struct Template Reference	1110
4.286.1 Detailed Description	1111
4.287 <code>__gnu_pbds::detail::pat_trie_base::_Node_citer<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc></code> Class Template Reference	1111
4.287.1 Detailed Description	1112
4.287.2 Member Typedef Documentation	1112
4.287.3 Member Function Documentation	1113
4.288 <code>__gnu_pbds::detail::pat_trie_base::_Node_iter<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc></code> Class Template Reference	1114
4.288.1 Detailed Description	1115
4.288.2 Member Typedef Documentation	1116
4.288.3 Member Function Documentation	1116
4.289 <code>__gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc></code> Class Template Reference	1117
4.289.1 Detailed Description	1120
4.289.2 Member Enumeration Documentation	1120
4.289.3 Member Function Documentation	1120
4.290 <code>__gnu_pbds::detail::probe_fn_base<_Alloc></code> Class Template Reference	1121
4.290.1 Detailed Description	1121
4.291 <code>__gnu_pbds::detail::ranged_hash_fn<Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false></code> Class Template Reference	1121
4.291.1 Detailed Description	1122

4.292 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference	1122
4.292.1 Detailed Description	1122
4.293 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference	1123
4.293.1 Detailed Description	1123
4.294 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference	1123
4.294.1 Detailed Description	1124
4.295 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference	1124
4.295.1 Detailed Description	1124
4.296 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference	1125
4.296.1 Detailed Description	1125
4.297 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference	1125
4.297.1 Detailed Description	1126
4.298 __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference	1126
4.298.1 Detailed Description	1129
4.298.2 Member Function Documentation	1129
4.299 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference	1130
4.299.1 Detailed Description	1131
4.300 __gnu_pbds::detail::rc< _Node, _Alloc > Class Template Reference	1131
4.300.1 Detailed Description	1132
4.301 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	1132
4.301.1 Detailed Description	1134
4.302 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference	1135
4.302.1 Detailed Description	1136
4.303 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference	1136
4.303.1 Detailed Description	1139
4.303.2 Member Function Documentation	1139
4.304 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference	1140
4.304.1 Detailed Description	1140
4.305 __gnu_pbds::detail::stored_data< _Tv, _Th > Struct Template Reference	1141
4.305.1 Detailed Description	1141
4.306 __gnu_pbds::detail::stored_data< _Tv, null_type > Struct Template Reference	1142

4.306.1 Detailed Description	1142
4.307 <code>__gnu_pbds::detail::stored_hash< _Th ></code> Struct Template Reference	1143
4.307.1 Detailed Description	1143
4.308 <code>__gnu_pbds::detail::stored_value< _Tv ></code> Struct Template Reference	1144
4.308.1 Detailed Description	1144
4.309 <code>__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits ></code> Struct Template Reference	1144
4.309.1 Detailed Description	1145
4.310 <code>__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc ></code> Class Template Reference	1145
4.310.1 Detailed Description	1147
4.311 <code>__gnu_pbds::detail::tree_metadata_helper< Node_Update, false ></code> Struct Template Reference	1147
4.311.1 Detailed Description	1148
4.312 <code>__gnu_pbds::detail::tree_metadata_helper< Node_Update, true ></code> Struct Template Reference	1148
4.312.1 Detailed Description	1148
4.313 <code>__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc ></code> Struct Template Reference	1148
4.313.1 Detailed Description	1148
4.314 <code>__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc ></code> Struct Template Reference	1149
4.314.1 Detailed Description	1149
4.314.2 Member Typedef Documentation	1149
4.315 <code>__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc ></code> Struct Template Reference	1150
4.315.1 Detailed Description	1152
4.315.2 Member Typedef Documentation	1152
4.316 <code>__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc ></code> Struct Template Reference	1153
4.316.1 Detailed Description	1155
4.316.2 Member Typedef Documentation	1155
4.317 <code>__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc ></code> Struct Template Reference	1155
4.317.1 Detailed Description	1156
4.317.2 Member Typedef Documentation	1156
4.318 <code>__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc ></code> Struct Template Reference	1156
4.318.1 Detailed Description	1158
4.318.2 Member Typedef Documentation	1158
4.319 <code>__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc ></code> Struct Template Reference	1159
4.319.1 Detailed Description	1161

4.319.2 Member Typedef Documentation	1161
4.320 <code>__gnu_pbds::detail::trie_metadata_helper< Node_Update, false ></code> Struct Template Reference	1161
4.320.1 Detailed Description	1161
4.321 <code>__gnu_pbds::detail::trie_metadata_helper< Node_Update, true ></code> Struct Template Reference	1162
4.321.1 Detailed Description	1162
4.322 <code>__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc ></code> Struct Template Reference	1162
4.322.1 Detailed Description	1162
4.323 <code>__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc ></code> Class Template Reference	1163
4.323.1 Detailed Description	1164
4.324 <code>__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc ></code> Struct Template Reference	1164
4.324.1 Detailed Description	1165
4.324.2 Member Typedef Documentation	1165
4.325 <code>__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc ></code> Struct Template Reference	1166
4.325.1 Detailed Description	1167
4.325.2 Member Typedef Documentation	1167
4.326 <code>__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false ></code> Struct Template Reference	1168
4.326.1 Detailed Description	1168
4.327 <code>__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true ></code> Struct Template Reference	1168
4.327.1 Detailed Description	1169
4.328 <code>__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false ></code> Struct Template Reference	1169
4.328.1 Detailed Description	1169
4.329 <code>__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true ></code> Struct Template Reference	1170
4.329.1 Detailed Description	1170
4.330 <code>__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash ></code> Struct Template Reference	1170
4.330.1 Detailed Description	1170
4.331 <code>__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash ></code> Struct Template Reference	1171
4.331.1 Detailed Description	1171
4.332 <code>__gnu_pbds::direct_mask_range_hashing< Size_Type ></code> Class Template Reference	1172
4.332.1 Detailed Description	1172
4.332.2 Member Function Documentation	1172
4.333 <code>__gnu_pbds::direct_mod_range_hashing< Size_Type ></code> Class Template Reference	1173
4.333.1 Detailed Description	1174
4.333.2 Member Function Documentation	1174
4.334 <code>__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc ></code> Class Template Reference	1174

4.334.1 Detailed Description	1175
4.334.2 Constructor & Destructor Documentation	1176
4.335 <code>__gnu_pbds::gp_hash_tag</code> Struct Reference	1179
4.335.1 Detailed Description	1179
4.336 <code>__gnu_pbds::hash_exponential_size_policy< Size_Type ></code> Class Template Reference	1179
4.336.1 Detailed Description	1180
4.336.2 Constructor & Destructor Documentation	1180
4.337 <code>__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type ></code> Class Template Reference	1180
4.337.1 Detailed Description	1181
4.337.2 Member Enumeration Documentation	1181
4.337.3 Constructor & Destructor Documentation	1181
4.337.4 Member Function Documentation	1182
4.338 <code>__gnu_pbds::hash_prime_size_policy</code> Class Reference	1182
4.338.1 Detailed Description	1183
4.338.2 Member Typedef Documentation	1183
4.338.3 Constructor & Destructor Documentation	1183
4.339 <code>__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type ></code> Class Template Reference	1183
4.339.1 Detailed Description	1184
4.339.2 Constructor & Destructor Documentation	1184
4.339.3 Member Function Documentation	1185
4.340 <code>__gnu_pbds::insert_error</code> Struct Reference	1186
4.340.1 Detailed Description	1187
4.340.2 Member Function Documentation	1187
4.341 <code>__gnu_pbds::join_error</code> Struct Reference	1187
4.341.1 Detailed Description	1188
4.341.2 Member Function Documentation	1188
4.342 <code>__gnu_pbds::linear_probe_fn< Size_Type ></code> Class Template Reference	1188
4.342.1 Detailed Description	1188
4.342.2 Member Function Documentation	1188
4.343 <code>__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc ></code> Class Template Reference	1189
4.343.1 Detailed Description	1189
4.343.2 Constructor & Destructor Documentation	1189
4.344 <code>__gnu_pbds::list_update_tag</code> Struct Reference	1190
4.344.1 Detailed Description	1190
4.345 <code>__gnu_pbds::lu_counter_policy< Max_Count, _Alloc ></code> Class Template Reference	1191

4.345.1 Detailed Description	1192
4.345.2 Member Typedef Documentation	1192
4.345.3 Member Enumeration Documentation	1192
4.345.4 Member Function Documentation	1192
4.346 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference	1193
4.346.1 Detailed Description	1193
4.346.2 Member Typedef Documentation	1193
4.346.3 Member Function Documentation	1193
4.347 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference	1194
4.347.1 Detailed Description	1194
4.348 __gnu_pbds::null_type Struct Reference	1195
4.348.1 Detailed Description	1195
4.349 __gnu_pbds::ov_tree_tag Struct Reference	1196
4.349.1 Detailed Description	1196
4.350 __gnu_pbds::pairing_heap_tag Struct Reference	1197
4.350.1 Detailed Description	1197
4.351 __gnu_pbds::pat_trie_tag Struct Reference	1198
4.351.1 Detailed Description	1198
4.352 __gnu_pbds::point_invalidation_guarantee Struct Reference	1199
4.352.1 Detailed Description	1199
4.353 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference	1199
4.353.1 Detailed Description	1200
4.354 __gnu_pbds::priority_queue_tag Struct Reference	1201
4.354.1 Detailed Description	1201
4.355 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference	1201
4.355.1 Detailed Description	1201
4.355.2 Member Function Documentation	1202
4.356 __gnu_pbds::range_invalidation_guarantee Struct Reference	1202
4.356.1 Detailed Description	1202
4.357 __gnu_pbds::rb_tree_tag Struct Reference	1203
4.357.1 Detailed Description	1203
4.358 __gnu_pbds::rc_binomial_heap_tag Struct Reference	1204
4.358.1 Detailed Description	1204
4.359 __gnu_pbds::resize_error Struct Reference	1205
4.359.1 Detailed Description	1205
4.359.2 Member Function Documentation	1205
4.360 __gnu_pbds::sample_probe_fn Class Reference	1206

4.360.1 Detailed Description	1206
4.360.2 Constructor & Destructor Documentation	1206
4.360.3 Member Function Documentation	1206
4.361 __gnu_pbds::sample_range_hashing Class Reference	1206
4.361.1 Detailed Description	1207
4.361.2 Member Typedef Documentation	1207
4.361.3 Constructor & Destructor Documentation	1207
4.361.4 Member Function Documentation	1207
4.362 __gnu_pbds::sample_ranged_hash_fn Class Reference	1208
4.362.1 Detailed Description	1208
4.362.2 Constructor & Destructor Documentation	1208
4.362.3 Member Function Documentation	1208
4.363 __gnu_pbds::sample_ranged_probe_fn Class Reference	1209
4.363.1 Detailed Description	1209
4.364 __gnu_pbds::sample_resize_policy Class Reference	1209
4.364.1 Detailed Description	1210
4.364.2 Member Typedef Documentation	1210
4.364.3 Constructor & Destructor Documentation	1210
4.364.4 Member Function Documentation	1210
4.365 __gnu_pbds::sample_resize_trigger Class Reference	1211
4.365.1 Detailed Description	1212
4.365.2 Member Typedef Documentation	1212
4.365.3 Constructor & Destructor Documentation	1212
4.365.4 Member Function Documentation	1212
4.366 __gnu_pbds::sample_size_policy Class Reference	1214
4.366.1 Detailed Description	1214
4.366.2 Member Typedef Documentation	1214
4.366.3 Constructor & Destructor Documentation	1214
4.366.4 Member Function Documentation	1215
4.367 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference	1215
4.367.1 Detailed Description	1215
4.368 __gnu_pbds::sample_trie_access_traits Struct Reference	1215
4.368.1 Detailed Description	1216
4.368.2 Member Typedef Documentation	1216
4.368.3 Member Function Documentation	1216

4.369 __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	1216
4.369.1 Detailed Description	1217
4.369.2 Constructor & Destructor Documentation	1217
4.369.3 Member Function Documentation	1217
4.370 __gnu_pbds::sample_update_policy Struct Reference	1217
4.370.1 Detailed Description	1217
4.370.2 Member Typedef Documentation	1218
4.370.3 Constructor & Destructor Documentation	1218
4.370.4 Member Function Documentation	1218
4.371 __gnu_pbds::sequence_tag Struct Reference	1219
4.371.1 Detailed Description	1219
4.372 __gnu_pbds::splay_tree_tag Struct Reference	1220
4.372.1 Detailed Description	1220
4.373 __gnu_pbds::string_tag Struct Reference	1221
4.373.1 Detailed Description	1221
4.374 __gnu_pbds::thin_heap_tag Struct Reference	1222
4.374.1 Detailed Description	1222
4.375 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference	1222
4.375.1 Detailed Description	1223
4.375.2 Member Typedef Documentation	1223
4.375.3 Constructor & Destructor Documentation	1223
4.376 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference	1224
4.376.1 Detailed Description	1226
4.376.2 Member Function Documentation	1226
4.377 __gnu_pbds::tree_tag Struct Reference	1227
4.377.1 Detailed Description	1227
4.378 __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference	1227
4.378.1 Detailed Description	1228
4.378.2 Member Typedef Documentation	1228
4.378.3 Constructor & Destructor Documentation	1229
4.379 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	1230
4.379.1 Detailed Description	1231
4.379.2 Member Function Documentation	1232
4.380 __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	1233

4.380.1 Detailed Description	1234
4.380.2 Member Typedef Documentation	1235
4.380.3 Member Function Documentation	1235
4.381 <code>__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc ></code> Struct Template Reference	1236
4.381.1 Detailed Description	1237
4.381.2 Member Typedef Documentation	1237
4.381.3 Member Function Documentation	1237
4.382 <code>__gnu_pbds::trie_tag</code> Struct Reference	1238
4.382.1 Detailed Description	1239
4.383 <code>__gnu_pbds::trivial_iterator_tag</code> Struct Reference	1239
4.383.1 Detailed Description	1239
4.384 <code>__gnu_profile::__container_size_info</code> Class Reference	1239
4.384.1 Detailed Description	1240
4.385 <code>__gnu_profile::__container_size_stack_info</code> Class Reference	1240
4.385.1 Detailed Description	1241
4.386 <code>__gnu_profile::__hashfunc_info</code> Class Reference	1241
4.386.1 Detailed Description	1242
4.387 <code>__gnu_profile::__hashfunc_stack_info</code> Class Reference	1242
4.387.1 Detailed Description	1243
4.388 <code>__gnu_profile::__list2vector_info</code> Class Reference	1243
4.388.1 Detailed Description	1244
4.389 <code>__gnu_profile::__map2umap_info</code> Class Reference	1245
4.389.1 Detailed Description	1245
4.390 <code>__gnu_profile::__map2umap_stack_info</code> Class Reference	1246
4.390.1 Detailed Description	1246
4.391 <code>__gnu_profile::__object_info_base</code> Class Reference	1247
4.391.1 Detailed Description	1247
4.392 <code>__gnu_profile::__reentrance_guard</code> Struct Reference	1248
4.392.1 Detailed Description	1248
4.393 <code>__gnu_profile::__stack_hash</code> Class Reference	1248
4.393.1 Detailed Description	1248
4.394 <code>__gnu_profile::__stack_info_base< __object_info ></code> Class Template Reference	1248
4.394.1 Detailed Description	1248
4.395 <code>__gnu_profile::__trace_base< __object_info, __stack_info ></code> Class Template Reference	1249
4.395.1 Detailed Description	1249
4.396 <code>__gnu_profile::__trace_container_size</code> Class Reference	1250

4.396.1 Detailed Description	1250
4.397 __gnu_profile::__trace_hash_func Class Reference	1251
4.397.1 Detailed Description	1251
4.398 __gnu_profile::__trace_hashtable_size Class Reference	1252
4.398.1 Detailed Description	1252
4.399 __gnu_profile::__trace_map2umap Class Reference	1253
4.399.1 Detailed Description	1253
4.400 __gnu_profile::__trace_vector_size Class Reference	1254
4.400.1 Detailed Description	1254
4.401 __gnu_profile::__trace_vector_to_list Class Reference	1255
4.401.1 Detailed Description	1256
4.402 __gnu_profile::__vector2list_info Class Reference	1256
4.402.1 Detailed Description	1257
4.403 __gnu_profile::__vector2list_stack_info Class Reference	1257
4.403.1 Detailed Description	1258
4.404 __gnu_profile::__warning_data Struct Reference	1258
4.404.1 Detailed Description	1258
4.405 __pool< _Thread > Class Template Reference	1259
4.405.1 Detailed Description	1259
4.406 __reflection_typelist< _Elements > Struct Template Reference	1259
4.406.1 Detailed Description	1259
4.407 _Bind< _Signature > Struct Template Reference	1259
4.407.1 Detailed Description	1259
4.408 _Bind_result< _Result, _Signature > Struct Template Reference	1259
4.408.1 Detailed Description	1259
4.409 _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code > Struct Template Reference	1260
4.409.1 Detailed Description	1260
4.410 _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _ Unique_keys > Struct Template Reference	1260
4.410.1 Detailed Description	1260
4.411 _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Tem- plate Reference	1260
4.411.1 Detailed Description	1260
4.412 _Hash_node< _Value, _Cache_hash_code > Struct Template Reference	1261
4.412.1 Detailed Description	1261
4.413 _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo > Struct Template Reference	1261
4.413.1 Detailed Description	1262

4.414	_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys > Struct Template Reference	1262
4.414.1	Detailed Description	1262
4.415	_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference	1262
4.415.1	Detailed Description	1262
4.416	_Mu< _Arg, _IsBindExp, _IsPlaceholder > Class Template Reference	1263
4.416.1	Detailed Description	1263
4.417	_Reference_wrapper_base_impl< _Unary, _Binary, _Tp > Struct Template Reference	1263
4.417.1	Detailed Description	1263
4.418	_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference	1264
4.418.1	Detailed Description	1264
4.419	_Tuple_impl< _Idx, _Elements > Struct Template Reference	1264
4.419.1	Detailed Description	1264
4.420	common_type< _Tp > Struct Template Reference	1264
4.420.1	Detailed Description	1265
4.421	const_iterator_ Class Reference	1265
4.421.1	Detailed Description	1266
4.421.2	Member Typedef Documentation	1266
4.421.3	Constructor & Destructor Documentation	1267
4.421.4	Member Function Documentation	1267
4.421.5	Member Data Documentation	1268
4.422	container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI > Struct Template Reference	1268
4.422.1	Detailed Description	1268
4.423	container_traits_base< _Tag > Struct Template Reference	1269
4.423.1	Detailed Description	1269
4.424	default_trie_access_traits< Key > Struct Template Reference	1269
4.424.1	Detailed Description	1269
4.425	entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw > Struct Template Reference	1270
4.425.1	Detailed Description	1270
4.426	entry_pred< _VTp, Pred, _Alloc, No_Throw > Struct Template Reference	1270
4.426.1	Detailed Description	1270
4.427	hash< _Tp > Struct Template Reference	1270
4.427.1	Detailed Description	1270
4.428	hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference	1271
4.428.1	Detailed Description	1271
4.429	hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference	1271

4.429.1 Detailed Description	1271
4.430iterator_ Class Reference	1272
4.430.1 Detailed Description	1273
4.430.2 Member Typedef Documentation	1273
4.430.3 Constructor & Destructor Documentation	1274
4.430.4 Member Function Documentation	1274
4.430.5 Member Data Documentation	1275
4.431owner_less< _Tp > Struct Template Reference	1275
4.431.1 Detailed Description	1275
4.432point_const_iterator_ Class Reference	1276
4.432.1 Detailed Description	1277
4.432.2 Member Typedef Documentation	1277
4.432.3 Constructor & Destructor Documentation	1277
4.432.4 Member Function Documentation	1278
4.433point_iterator_ Class Reference	1279
4.433.1 Detailed Description	1279
4.433.2 Member Typedef Documentation	1279
4.433.3 Constructor & Destructor Documentation	1280
4.433.4 Member Function Documentation	1280
4.434ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference	1281
4.434.1 Detailed Description	1281
4.435ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference	1282
4.435.1 Detailed Description	1282
4.436result_of< _Signature > Class Template Reference	1282
4.436.1 Detailed Description	1282
4.437std::__atomic_base< _ITp > Struct Template Reference	1283
4.437.1 Detailed Description	1285
4.438std::__atomic_base< _PTp * > Struct Template Reference	1285
4.438.1 Detailed Description	1286
4.439std::__atomic_flag_base Struct Reference	1286
4.439.1 Detailed Description	1287
4.440std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference	1287
4.440.1 Detailed Description	1288
4.440.2 Member Function Documentation	1288
4.441std::__ctype_abstract_base< _CharT > Class Template Reference	1291
4.441.1 Detailed Description	1292

4.441.2 Member Typedef Documentation	1293
4.441.3 Member Function Documentation	1293
4.442std::__debug::bitset< _Nb > Class Template Reference	1302
4.442.1 Detailed Description	1304
4.443std::__debug::deque< _Tp, _Allocator > Class Template Reference	1304
4.443.1 Detailed Description	1307
4.443.2 Member Function Documentation	1307
4.443.3 Member Data Documentation	1308
4.444std::__debug::forward_list< _Tp, _Alloc > Class Template Reference	1309
4.444.1 Detailed Description	1311
4.444.2 Member Function Documentation	1311
4.444.3 Member Data Documentation	1313
4.445std::__debug::list< _Tp, _Allocator > Class Template Reference	1314
4.445.1 Detailed Description	1316
4.445.2 Member Function Documentation	1317
4.445.3 Member Data Documentation	1318
4.446std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1319
4.446.1 Detailed Description	1321
4.446.2 Member Function Documentation	1321
4.446.3 Member Data Documentation	1323
4.447std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1324
4.447.1 Detailed Description	1326
4.447.2 Member Function Documentation	1326
4.447.3 Member Data Documentation	1328
4.448std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	1329
4.448.1 Detailed Description	1331
4.448.2 Member Function Documentation	1331
4.448.3 Member Data Documentation	1333
4.449std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference	1334
4.449.1 Detailed Description	1336
4.449.2 Member Function Documentation	1336
4.449.3 Member Data Documentation	1338
4.450std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1338
4.450.1 Detailed Description	1340
4.450.2 Member Function Documentation	1341
4.450.3 Member Data Documentation	1342
4.451std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1343

4.451.1 Detailed Description	1345
4.451.2 Member Function Documentation	1345
4.451.3 Member Data Documentation	1347
4.452std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1348
4.452.1 Detailed Description	1350
4.452.2 Member Function Documentation	1350
4.452.3 Member Data Documentation	1352
4.453std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1353
4.453.1 Detailed Description	1355
4.453.2 Member Function Documentation	1355
4.453.3 Member Data Documentation	1357
4.454std::__debug::vector< _Tp, _Allocator > Class Template Reference	1358
4.454.1 Detailed Description	1361
4.454.2 Constructor & Destructor Documentation	1361
4.454.3 Member Function Documentation	1361
4.454.4 Member Data Documentation	1362
4.455std::__detail::_Automaton Class Reference	1363
4.455.1 Detailed Description	1363
4.456std::__detail::_Before_begin< _NodeAlloc > Struct Template Reference	1364
4.456.1 Detailed Description	1364
4.457std::__detail::_CharMatcher< _InIterT, _TraitsT > Struct Template Reference	1364
4.457.1 Detailed Description	1365
4.458std::__detail::_Compiler< _InIter, _TraitsT > Class Template Reference	1365
4.458.1 Detailed Description	1365
4.459std::__detail::_Default_ranged_hash Struct Reference	1366
4.459.1 Detailed Description	1366
4.460std::__detail::_EndTagger< _FwdIterT, _TraitsT > Struct Template Reference	1366
4.460.1 Detailed Description	1366
4.461std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false > Struct Template Reference	1366
4.461.1 Detailed Description	1366
4.462std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true > Struct Template Reference	1367
4.462.1 Detailed Description	1367
4.463std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1367
4.463.1 Detailed Description	1368

4.464std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1368
4.464.1 Detailed Description	1368
4.465std::__detail::Equality_base Struct Reference	1369
4.465.1 Detailed Description	1369
4.466std::__detail::Grep_matcher Class Reference	1369
4.466.1 Detailed Description	1369
4.467std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false > Struct Template Reference	1370
4.467.1 Detailed Description	1371
4.468std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true > Struct Template Reference	1371
4.468.1 Detailed Description	1372
4.469std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference	1372
4.469.1 Detailed Description	1373
4.470std::__detail::Hash_node< _Value, false > Struct Template Reference	1373
4.470.1 Detailed Description	1374
4.471std::__detail::Hash_node< _Value, true > Struct Template Reference	1374
4.471.1 Detailed Description	1375
4.472std::__detail::Hash_node_base Struct Reference	1375
4.472.1 Detailed Description	1375
4.473std::__detail::Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits > Struct Template Reference	1376
4.473.1 Detailed Description	1377
4.474std::__detail::Hashtable_ebo_helper< _Nm, _Tp, false > Struct Template Reference	1377
4.474.1 Detailed Description	1377
4.475std::__detail::Hashtable_ebo_helper< _Nm, _Tp, true > Struct Template Reference	1377
4.475.1 Detailed Description	1378
4.476std::__detail::Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys > Struct Template Reference	1378
4.476.1 Detailed Description	1378
4.477std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys > Struct Template Reference	1379
4.477.1 Detailed Description	1380
4.478std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false > Struct Template Reference	1380
4.478.1 Detailed Description	1381
4.479std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true > Struct Template Reference	1381

4.479.1 Detailed Description	1382
4.480std::__detail::Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference	1383
4.480.1 Detailed Description	1384
4.481std::__detail::_List_node_base Struct Reference	1384
4.481.1 Detailed Description	1385
4.482std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > Struct Template Reference	1385
4.482.1 Detailed Description	1385
4.483std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > Struct Template Reference	1386
4.483.1 Detailed Description	1386
4.484std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference	1387
4.484.1 Detailed Description	1388
4.485std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true > Struct Template Reference	1388
4.485.1 Detailed Description	1389
4.486std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1389
4.486.1 Detailed Description	1389
4.487std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1390
4.487.1 Detailed Description	1390
4.488std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1390
4.488.1 Detailed Description	1390
4.489std::__detail::_Mod_range_hashing Struct Reference	1391
4.489.1 Detailed Description	1391
4.490std::__detail::_Nfa Class Reference	1392
4.490.1 Detailed Description	1395
4.490.2 Member Function Documentation	1395
4.491std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1405
4.491.1 Detailed Description	1406
4.492std::__detail::_Node_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1406
4.492.1 Detailed Description	1407
4.493std::__detail::_Node_iterator_base< _Value, _Cache_hash_code > Struct Template Reference	1407
4.493.1 Detailed Description	1408
4.494std::__detail::_PatternCursor Struct Reference	1408

4.494.1 Detailed Description	1408
4.495std::__detail::__Prime_rehash_policy Struct Reference	1409
4.495.1 Detailed Description	1409
4.496std::__detail::__RangeMatcher<_InIterT, _TraitsT> Struct Template Reference	1409
4.496.1 Detailed Description	1410
4.497std::__detail::__Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits> Struct Template Reference	1410
4.497.1 Detailed Description	1410
4.498std::__detail::__Results Struct Reference	1411
4.498.1 Detailed Description	1411
4.499std::__detail::__Scanner<_InputIterator> Class Template Reference	1412
4.499.1 Detailed Description	1413
4.499.2 Member Enumeration Documentation	1413
4.500std::__detail::__Scanner_base Struct Reference	1414
4.500.1 Detailed Description	1414
4.501std::__detail::__SpecializedCursor<_FwdIterT> Class Template Reference	1415
4.501.1 Detailed Description	1415
4.502std::__detail::__SpecializedResults<_FwdIterT, _Alloc> Class Template Reference	1416
4.502.1 Detailed Description	1416
4.503std::__detail::__StartTagger<_FwdIterT, _TraitsT> Struct Template Reference	1416
4.503.1 Detailed Description	1417
4.504std::__detail::__State Struct Reference	1417
4.504.1 Detailed Description	1417
4.505std::__detail::__StateSeq Class Reference	1418
4.505.1 Detailed Description	1418
4.506std::__exception_ptr::exception_ptr Class Reference	1418
4.506.1 Detailed Description	1419
4.507std::__has_iterator_category_helper<_Tp> Class Template Reference	1419
4.507.1 Detailed Description	1419
4.508std::__is_location_invariant<_Tp> Struct Template Reference	1419
4.508.1 Detailed Description	1420
4.509std::__numeric_limits_base Struct Reference	1420
4.509.1 Detailed Description	1421
4.509.2 Member Data Documentation	1421
4.510std::__parallel::__CRandNumber<_MustBeInt> Struct Template Reference	1424
4.510.1 Detailed Description	1424
4.511std::__profile::bitset<_Nb> Class Template Reference	1424

4.511.1 Detailed Description	1426
4.512std::__profile::deque< _Tp, _Allocator > Class Template Reference	1426
4.512.1 Detailed Description	1428
4.513std::__profile::forward_list< _Tp, _Alloc > Class Template Reference	1428
4.513.1 Detailed Description	1428
4.514std::__profile::list< _Tp, _Allocator > Class Template Reference	1429
4.514.1 Detailed Description	1431
4.515std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1431
4.515.1 Detailed Description	1433
4.516std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1433
4.516.1 Detailed Description	1434
4.517std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference	1435
4.517.1 Detailed Description	1436
4.518std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference	1436
4.518.1 Detailed Description	1438
4.519std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1438
4.519.1 Detailed Description	1439
4.520std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1439
4.520.1 Detailed Description	1441
4.521std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1441
4.521.1 Detailed Description	1442
4.522std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference	1442
4.522.1 Detailed Description	1443
4.523std::_Base_bitset< _Nw > Struct Template Reference	1443
4.523.1 Detailed Description	1444
4.523.2 Member Data Documentation	1444
4.524std::_Base_bitset< 0 > Struct Template Reference	1445
4.524.1 Detailed Description	1445
4.525std::_Base_bitset< 1 > Struct Template Reference	1446
4.525.1 Detailed Description	1447
4.526std::_Deque_base< _Tp, _Alloc > Class Template Reference	1447
4.526.1 Detailed Description	1448
4.526.2 Member Function Documentation	1448
4.527std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference	1449
4.527.1 Detailed Description	1450
4.527.2 Member Function Documentation	1450
4.528std::_Derives_from_binary_function< _Tp > Struct Template Reference	1451

4.528.1 Detailed Description	1451
4.529std::_Derives_from_unary_function< _Tp > Struct Template Reference	1451
4.529.1 Detailed Description	1451
4.530std::_Function_base Class Reference	1452
4.530.1 Detailed Description	1452
4.531std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference	1453
4.531.1 Detailed Description	1454
4.532std::_Fwd_list_const_iterator< _Tp > Struct Template Reference	1454
4.532.1 Detailed Description	1455
4.533std::_Fwd_list_iterator< _Tp > Struct Template Reference	1455
4.533.1 Detailed Description	1455
4.534std::_Fwd_list_node< _Tp > Struct Template Reference	1456
4.534.1 Detailed Description	1456
4.535std::_Fwd_list_node_base Struct Reference	1457
4.535.1 Detailed Description	1457
4.536std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Class Template Reference	1458
4.536.1 Detailed Description	1460
4.537std::_List_base< _Tp, _Alloc > Class Template Reference	1462
4.537.1 Detailed Description	1463
4.538std::_List_const_iterator< _Tp > Struct Template Reference	1463
4.538.1 Detailed Description	1464
4.539std::_List_iterator< _Tp > Struct Template Reference	1464
4.539.1 Detailed Description	1465
4.540std::_List_node< _Tp > Struct Template Reference	1465
4.540.1 Detailed Description	1466
4.540.2 Member Data Documentation	1466
4.541std::_Maybe_get_result_type< _Has_result_type, _Functor > Struct Template Reference	1466
4.541.1 Detailed Description	1467
4.542std::_Maybe_unary_or_binary_function< _Res, _ArgTypes > Struct Template Reference	1467
4.542.1 Detailed Description	1467
4.543std::_Maybe_unary_or_binary_function< _Res, _T1 > Struct Template Reference	1468
4.543.1 Detailed Description	1468
4.543.2 Member Typedef Documentation	1468
4.544std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 > Struct Template Reference	1469
4.544.1 Detailed Description	1469
4.544.2 Member Typedef Documentation	1469

4.545std::_Maybe_wrap_member_pointer< _Tp > Struct Template Reference	1470
4.545.1 Detailed Description	1470
4.546std::_Maybe_wrap_member_pointer< _Tp_Class::* > Struct Template Reference	1470
4.546.1 Detailed Description	1471
4.547std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const > Class Template Reference	1471
4.547.1 Detailed Description	1472
4.548std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile > Class Template Reference	1472
4.548.1 Detailed Description	1473
4.549std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile > Class Template Reference	1473
4.549.1 Detailed Description	1474
4.550std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) > Class Template Reference	1474
4.550.1 Detailed Description	1475
4.551std::_Mu< _Arg, false, false > Class Template Reference	1475
4.551.1 Detailed Description	1475
4.552std::_Mu< _Arg, false, true > Class Template Reference	1475
4.552.1 Detailed Description	1476
4.553std::_Mu< _Arg, true, false > Class Template Reference	1476
4.553.1 Detailed Description	1476
4.554std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference	1476
4.554.1 Detailed Description	1476
4.555std::_Placeholder< _Num > Struct Template Reference	1477
4.555.1 Detailed Description	1477
4.556std::_Reference_wrapper_base< _Tp > Struct Template Reference	1477
4.556.1 Detailed Description	1477
4.557std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference	1477
4.557.1 Detailed Description	1478
4.558std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference	1478
4.558.1 Detailed Description	1479
4.559std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference	1479
4.559.1 Detailed Description	1479
4.560std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference	1479
4.560.1 Detailed Description	1480
4.560.2 Constructor & Destructor Documentation	1480
4.560.3 Member Function Documentation	1480
4.561std::_Tuple_impl< _Idx > Struct Template Reference	1481
4.561.1 Detailed Description	1481
4.562std::_Tuple_impl< _Idx, _Head, _Tail... > Struct Template Reference	1482

4.562.1 Detailed Description	1484
4.563std::_Vector_base< _Tp, _Alloc > Struct Template Reference	1484
4.563.1 Detailed Description	1485
4.564std::_Weak_result_type< _Functor > Struct Template Reference	1485
4.564.1 Detailed Description	1485
4.565std::_Weak_result_type_impl< _Functor > Struct Template Reference	1486
4.565.1 Detailed Description	1486
4.566std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)> Struct Template Reference	1486
4.566.1 Detailed Description	1486
4.567std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)> Struct Template Reference	1486
4.567.1 Detailed Description	1486
4.568std::_Weak_result_type_impl< _Res(_ArgTypes...)> Struct Template Reference	1487
4.568.1 Detailed Description	1487
4.569std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const > Struct Template Reference	1487
4.569.1 Detailed Description	1487
4.570std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile > Struct Template Reference	1487
4.570.1 Detailed Description	1487
4.571std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile > Struct Template Reference	1488
4.571.1 Detailed Description	1488
4.572std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)> Struct Template Reference	1488
4.572.1 Detailed Description	1488
4.573std::add_const< _Tp > Struct Template Reference	1488
4.573.1 Detailed Description	1489
4.574std::add_cv< _Tp > Struct Template Reference	1489
4.574.1 Detailed Description	1489
4.575std::add_lvalue_reference< _Tp > Struct Template Reference	1489
4.575.1 Detailed Description	1489
4.576std::add_pointer< _Tp > Struct Template Reference	1490
4.576.1 Detailed Description	1490
4.577std::add_rvalue_reference< _Tp > Struct Template Reference	1490
4.577.1 Detailed Description	1490
4.578std::add_volatile< _Tp > Struct Template Reference	1490
4.578.1 Detailed Description	1490
4.579std::adopt_lock_t Struct Reference	1491
4.579.1 Detailed Description	1491
4.580std::aligned_storage< _Len, _Align > Struct Template Reference	1491
4.580.1 Detailed Description	1491

4.581std::alignment_of< _Tp > Struct Template Reference	1492
4.581.1 Detailed Description	1492
4.582std::allocator< _Tp > Struct Template Reference	1493
4.582.1 Detailed Description	1494
4.583std::allocator< void > Class Template Reference	1494
4.583.1 Detailed Description	1494
4.584std::allocator_arg_t Struct Reference	1495
4.584.1 Detailed Description	1495
4.585std::allocator_traits< _Alloc > Struct Template Reference	1495
4.585.1 Detailed Description	1496
4.585.2 Member Typedef Documentation	1496
4.585.3 Member Function Documentation	1498
4.586std::array< _Tp, _Nm > Struct Template Reference	1500
4.586.1 Detailed Description	1501
4.587std::atomic< _Tp > Struct Template Reference	1501
4.587.1 Detailed Description	1502
4.588std::atomic< _Tp * > Struct Template Reference	1502
4.588.1 Detailed Description	1503
4.589std::atomic< bool > Struct Template Reference	1504
4.589.1 Detailed Description	1505
4.590std::atomic< char > Struct Template Reference	1505
4.590.1 Detailed Description	1507
4.591std::atomic< char16_t > Struct Template Reference	1507
4.591.1 Detailed Description	1509
4.592std::atomic< char32_t > Struct Template Reference	1509
4.592.1 Detailed Description	1511
4.593std::atomic< int > Struct Template Reference	1511
4.593.1 Detailed Description	1513
4.594std::atomic< long > Struct Template Reference	1513
4.594.1 Detailed Description	1515
4.595std::atomic< long long > Struct Template Reference	1515
4.595.1 Detailed Description	1517
4.596std::atomic< short > Struct Template Reference	1517
4.596.1 Detailed Description	1519
4.597std::atomic< signed char > Struct Template Reference	1519
4.597.1 Detailed Description	1521
4.598std::atomic< unsigned char > Struct Template Reference	1521

4.598.1 Detailed Description	1523
4.599std::atomic< unsigned int > Struct Template Reference	1523
4.599.1 Detailed Description	1525
4.600std::atomic< unsigned long > Struct Template Reference	1525
4.600.1 Detailed Description	1527
4.601std::atomic< unsigned long long > Struct Template Reference	1527
4.601.1 Detailed Description	1529
4.602std::atomic< unsigned short > Struct Template Reference	1529
4.602.1 Detailed Description	1531
4.603std::atomic< wchar_t > Struct Template Reference	1531
4.603.1 Detailed Description	1533
4.604std::atomic_bool Struct Reference	1533
4.604.1 Detailed Description	1534
4.605std::atomic_flag Struct Reference	1534
4.605.1 Detailed Description	1535
4.606std::auto_ptr< _Tp > Class Template Reference	1535
4.606.1 Detailed Description	1535
4.606.2 Member Typedef Documentation	1536
4.606.3 Constructor & Destructor Documentation	1536
4.606.4 Member Function Documentation	1537
4.607std::auto_ptr_ref< _Tp1 > Struct Template Reference	1539
4.607.1 Detailed Description	1539
4.608std::back_insert_iterator< _Container > Class Template Reference	1540
4.608.1 Detailed Description	1541
4.608.2 Member Typedef Documentation	1541
4.608.3 Constructor & Destructor Documentation	1541
4.608.4 Member Function Documentation	1542
4.609std::bad_alloc Class Reference	1543
4.609.1 Detailed Description	1543
4.609.2 Member Function Documentation	1543
4.610std::bad_cast Class Reference	1544
4.610.1 Detailed Description	1544
4.610.2 Member Function Documentation	1544
4.611std::bad_exception Class Reference	1545
4.611.1 Detailed Description	1545
4.611.2 Member Function Documentation	1545
4.612std::bad_function_call Class Reference	1546

4.612.1 Detailed Description	1546
4.612.2 Member Function Documentation	1546
4.613std::bad_typeid Class Reference	1547
4.613.1 Detailed Description	1547
4.613.2 Member Function Documentation	1547
4.614std::bad_weak_ptr Class Reference	1548
4.614.1 Detailed Description	1548
4.614.2 Member Function Documentation	1548
4.615std::basic_filebuf< _CharT, _Traits > Class Template Reference	1549
4.615.1 Detailed Description	1551
4.615.2 Constructor & Destructor Documentation	1552
4.615.3 Member Function Documentation	1552
4.615.4 Member Data Documentation	1566
4.616std::basic_fstream< _CharT, _Traits > Class Template Reference	1570
4.616.1 Detailed Description	1576
4.616.2 Member Typedef Documentation	1576
4.616.3 Member Enumeration Documentation	1578
4.616.4 Constructor & Destructor Documentation	1578
4.616.5 Member Function Documentation	1579
4.616.6 Member Data Documentation	1613
4.617std::basic_ifstream< _CharT, _Traits > Class Template Reference	1618
4.617.1 Detailed Description	1623
4.617.2 Member Typedef Documentation	1623
4.617.3 Member Enumeration Documentation	1625
4.617.4 Constructor & Destructor Documentation	1625
4.617.5 Member Function Documentation	1626
4.617.6 Member Data Documentation	1653
4.618std::basic_ios< _CharT, _Traits > Class Template Reference	1658
4.618.1 Detailed Description	1661
4.618.2 Member Typedef Documentation	1661
4.618.3 Member Enumeration Documentation	1664
4.618.4 Constructor & Destructor Documentation	1664
4.618.5 Member Function Documentation	1665
4.618.6 Member Data Documentation	1676
4.619std::basic_iostream< _CharT, _Traits > Class Template Reference	1681
4.619.1 Detailed Description	1687
4.619.2 Member Typedef Documentation	1687

4.619.3 Member Enumeration Documentation	1689
4.619.4 Constructor & Destructor Documentation	1689
4.619.5 Member Function Documentation	1689
4.619.6 Member Data Documentation	1722
4.620std::basic_istream< _CharT, _Traits > Class Template Reference	1728
4.620.1 Detailed Description	1732
4.620.2 Member Typedef Documentation	1732
4.620.3 Member Enumeration Documentation	1734
4.620.4 Constructor & Destructor Documentation	1735
4.620.5 Member Function Documentation	1735
4.620.6 Member Data Documentation	1761
4.621std::basic_istream< _CharT, _Traits >::sentry Class Reference	1766
4.621.1 Detailed Description	1766
4.621.2 Member Typedef Documentation	1766
4.621.3 Constructor & Destructor Documentation	1767
4.621.4 Member Function Documentation	1767
4.622std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference	1768
4.622.1 Detailed Description	1773
4.622.2 Member Typedef Documentation	1773
4.622.3 Member Enumeration Documentation	1775
4.622.4 Constructor & Destructor Documentation	1775
4.622.5 Member Function Documentation	1776
4.622.6 Member Data Documentation	1802
4.623std::basic_ofstream< _CharT, _Traits > Class Template Reference	1807
4.623.1 Detailed Description	1811
4.623.2 Member Typedef Documentation	1811
4.623.3 Member Enumeration Documentation	1813
4.623.4 Constructor & Destructor Documentation	1814
4.623.5 Member Function Documentation	1814
4.623.6 Member Data Documentation	1835
4.624std::basic_ostream< _CharT, _Traits > Class Template Reference	1840
4.624.1 Detailed Description	1844
4.624.2 Member Typedef Documentation	1844
4.624.3 Member Enumeration Documentation	1846
4.624.4 Constructor & Destructor Documentation	1847
4.624.5 Member Function Documentation	1847
4.624.6 Member Data Documentation	1866

4.625std::basic_ostream<_CharT, _Traits >::sentry Class Reference	1871
4.625.1 Detailed Description	1871
4.625.2 Constructor & Destructor Documentation	1871
4.625.3 Member Function Documentation	1871
4.626std::basic_ostringstream<_CharT, _Traits, _Alloc > Class Template Reference	1872
4.626.1 Detailed Description	1877
4.626.2 Member Typedef Documentation	1877
4.626.3 Member Enumeration Documentation	1879
4.626.4 Constructor & Destructor Documentation	1879
4.626.5 Member Function Documentation	1880
4.626.6 Member Data Documentation	1899
4.627std::basic_regex<_Ch_type, _Rx_traits > Class Template Reference	1904
4.627.1 Detailed Description	1905
4.627.2 Constructor & Destructor Documentation	1905
4.627.3 Member Function Documentation	1908
4.628std::basic_streambuf<_CharT, _Traits > Class Template Reference	1912
4.628.1 Detailed Description	1914
4.628.2 Member Typedef Documentation	1915
4.628.3 Constructor & Destructor Documentation	1916
4.628.4 Member Function Documentation	1916
4.628.5 Member Data Documentation	1929
4.629std::basic_string<_CharT, _Traits, _Alloc > Class Template Reference	1930
4.629.1 Detailed Description	1934
4.629.2 Constructor & Destructor Documentation	1935
4.629.3 Member Function Documentation	1937
4.629.4 Member Data Documentation	1974
4.630std::basic_stringbuf<_CharT, _Traits, _Alloc > Class Template Reference	1975
4.630.1 Detailed Description	1977
4.630.2 Constructor & Destructor Documentation	1977
4.630.3 Member Function Documentation	1978
4.630.4 Member Data Documentation	1990
4.631std::basic_stringstream<_CharT, _Traits, _Alloc > Class Template Reference	1992
4.631.1 Detailed Description	1998
4.631.2 Member Typedef Documentation	1998
4.631.3 Member Enumeration Documentation	2000
4.631.4 Constructor & Destructor Documentation	2001
4.631.5 Member Function Documentation	2001

4.631.6 Member Data Documentation	2035
4.632std::bernoulli_distribution Class Reference	2040
4.632.1 Detailed Description	2040
4.632.2 Member Typedef Documentation	2041
4.632.3 Constructor & Destructor Documentation	2041
4.632.4 Member Function Documentation	2041
4.632.5 Friends And Related Function Documentation	2042
4.633std::bernoulli_distribution::param_type Struct Reference	2042
4.633.1 Detailed Description	2042
4.634std::bidirectional_iterator_tag Struct Reference	2043
4.634.1 Detailed Description	2043
4.635std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference	2044
4.635.1 Detailed Description	2045
4.635.2 Member Typedef Documentation	2045
4.636std::binary_negate< _Predicate > Class Template Reference	2045
4.636.1 Detailed Description	2046
4.636.2 Member Typedef Documentation	2046
4.637std::binder1st< _Operation > Class Template Reference	2047
4.637.1 Detailed Description	2047
4.637.2 Member Typedef Documentation	2048
4.638std::binder2nd< _Operation > Class Template Reference	2048
4.638.1 Detailed Description	2049
4.638.2 Member Typedef Documentation	2049
4.639std::binomial_distribution< _IntType > Class Template Reference	2049
4.639.1 Detailed Description	2050
4.639.2 Member Typedef Documentation	2051
4.639.3 Member Function Documentation	2051
4.639.4 Friends And Related Function Documentation	2052
4.640std::binomial_distribution< _IntType >::param_type Struct Reference	2053
4.640.1 Detailed Description	2053
4.641std::cauchy_distribution< _RealType > Class Template Reference	2053
4.641.1 Detailed Description	2054
4.641.2 Member Typedef Documentation	2054
4.641.3 Member Function Documentation	2055
4.641.4 Friends And Related Function Documentation	2055
4.642std::cauchy_distribution< _RealType >::param_type Struct Reference	2056
4.642.1 Detailed Description	2056

4.643std::char_traits< _CharT > Struct Template Reference	2057
4.643.1 Detailed Description	2058
4.644std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference	2058
4.644.1 Detailed Description	2059
4.645std::char_traits< char > Struct Template Reference	2059
4.645.1 Detailed Description	2059
4.646std::char_traits< wchar_t > Struct Template Reference	2060
4.646.1 Detailed Description	2060
4.647std::chi_squared_distribution< _RealType > Class Template Reference	2060
4.647.1 Detailed Description	2061
4.647.2 Member Typedef Documentation	2061
4.647.3 Member Function Documentation	2062
4.647.4 Friends And Related Function Documentation	2063
4.648std::chi_squared_distribution< _RealType >::param_type Struct Reference	2063
4.648.1 Detailed Description	2064
4.649std::chrono::duration< _Rep, _Period > Struct Template Reference	2064
4.649.1 Detailed Description	2065
4.650std::chrono::duration_values< _Rep > Struct Template Reference	2065
4.650.1 Detailed Description	2065
4.651std::chrono::system_clock Struct Reference	2066
4.651.1 Detailed Description	2066
4.652std::chrono::time_point< _Clock, _Dur > Struct Template Reference	2066
4.652.1 Detailed Description	2067
4.653std::chrono::treat_as_floating_point< _Rep > Struct Template Reference	2067
4.653.1 Detailed Description	2067
4.654std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference	2068
4.654.1 Detailed Description	2069
4.654.2 Member Function Documentation	2069
4.655std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference	2072
4.655.1 Detailed Description	2073
4.655.2 Member Function Documentation	2073
4.656std::codecvt< char, char, mbstate_t > Class Template Reference	2075
4.656.1 Detailed Description	2077
4.656.2 Member Function Documentation	2077
4.657std::codecvt< wchar_t, char, mbstate_t > Class Template Reference	2079
4.657.1 Detailed Description	2080
4.657.2 Member Function Documentation	2080

4.658std::codecvt_base Class Reference	2083
4.658.1 Detailed Description	2083
4.659std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	2084
4.659.1 Detailed Description	2085
4.659.2 Member Function Documentation	2085
4.660std::collate< _CharT > Class Template Reference	2088
4.660.1 Detailed Description	2089
4.660.2 Member Typedef Documentation	2089
4.660.3 Constructor & Destructor Documentation	2089
4.660.4 Member Function Documentation	2090
4.660.5 Member Data Documentation	2092
4.661std::collate_byname< _CharT > Class Template Reference	2093
4.661.1 Detailed Description	2094
4.661.2 Member Typedef Documentation	2094
4.661.3 Member Function Documentation	2094
4.661.4 Member Data Documentation	2097
4.662std::complex< _Tp > Struct Template Reference	2097
4.662.1 Detailed Description	2098
4.662.2 Member Typedef Documentation	2098
4.662.3 Constructor & Destructor Documentation	2098
4.662.4 Member Function Documentation	2098
4.663std::complex< double > Struct Template Reference	2099
4.663.1 Detailed Description	2099
4.664std::complex< float > Struct Template Reference	2099
4.664.1 Detailed Description	2100
4.665std::complex< long double > Struct Template Reference	2100
4.665.1 Detailed Description	2101
4.666std::condition_variable Class Reference	2101
4.666.1 Detailed Description	2101
4.667std::condition_variable_any Class Reference	2102
4.667.1 Detailed Description	2102
4.668std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference	2102
4.668.1 Detailed Description	2102
4.669std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2103
4.669.1 Detailed Description	2103
4.669.2 Member Typedef Documentation	2103
4.670std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2104

4.670.1 Detailed Description	2105
4.670.2 Member Typedef Documentation	2105
4.671std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2106
4.671.1 Detailed Description	2106
4.671.2 Member Typedef Documentation	2106
4.672std::const_mem_fun_t< _Ret, _Tp > Class Template Reference	2107
4.672.1 Detailed Description	2107
4.672.2 Member Typedef Documentation	2108
4.673std::ctype< _CharT > Class Template Reference	2108
4.673.1 Detailed Description	2110
4.673.2 Member Function Documentation	2110
4.673.3 Member Data Documentation	2119
4.674std::ctype< char > Class Template Reference	2120
4.674.1 Detailed Description	2122
4.674.2 Member Typedef Documentation	2122
4.674.3 Constructor & Destructor Documentation	2122
4.674.4 Member Function Documentation	2122
4.674.5 Member Data Documentation	2130
4.675std::ctype< wchar_t > Class Template Reference	2131
4.675.1 Detailed Description	2133
4.675.2 Member Typedef Documentation	2133
4.675.3 Constructor & Destructor Documentation	2133
4.675.4 Member Function Documentation	2134
4.675.5 Member Data Documentation	2143
4.676std::ctype_base Struct Reference	2143
4.676.1 Detailed Description	2144
4.677std::ctype_byname< _CharT > Class Template Reference	2144
4.677.1 Detailed Description	2146
4.677.2 Member Function Documentation	2146
4.677.3 Member Data Documentation	2155
4.678std::ctype_byname< char > Class Template Reference	2156
4.678.1 Detailed Description	2158
4.678.2 Member Typedef Documentation	2158
4.678.3 Member Function Documentation	2158
4.678.4 Member Data Documentation	2166
4.679std::decay< _Tp > Class Template Reference	2166
4.679.1 Detailed Description	2166

4.680	std::decimal::decimal128 Class Reference	2166
4.680.1	Detailed Description	2168
4.680.2	Constructor & Destructor Documentation	2168
4.681	std::decimal::decimal32 Class Reference	2168
4.681.1	Detailed Description	2169
4.681.2	Constructor & Destructor Documentation	2169
4.682	std::decimal::decimal64 Class Reference	2169
4.682.1	Detailed Description	2171
4.682.2	Constructor & Destructor Documentation	2171
4.683	std::default_delete<_Tp> Struct Template Reference	2171
4.683.1	Detailed Description	2171
4.684	std::default_delete<_Tp[]> Struct Template Reference	2171
4.684.1	Detailed Description	2172
4.685	std::defer_lock_t Struct Reference	2172
4.685.1	Detailed Description	2172
4.686	std::deque<_Tp, _Alloc> Class Template Reference	2172
4.686.1	Detailed Description	2176
4.686.2	Constructor & Destructor Documentation	2177
4.686.3	Member Function Documentation	2179
4.687	std::discard_block_engine<_RandomNumberEngine, __p, __r> Class Template Reference	2194
4.687.1	Detailed Description	2195
4.687.2	Member Typedef Documentation	2195
4.687.3	Constructor & Destructor Documentation	2195
4.687.4	Member Function Documentation	2196
4.687.5	Friends And Related Function Documentation	2197
4.688	std::discrete_distribution<_IntType> Class Template Reference	2198
4.688.1	Detailed Description	2199
4.688.2	Member Typedef Documentation	2200
4.688.3	Member Function Documentation	2200
4.688.4	Friends And Related Function Documentation	2201
4.689	std::discrete_distribution<_IntType>::param_type Struct Reference	2202
4.689.1	Detailed Description	2202
4.690	std::divides<_Tp> Struct Template Reference	2203
4.690.1	Detailed Description	2203
4.690.2	Member Typedef Documentation	2203
4.691	std::domain_error Class Reference	2204
4.691.1	Detailed Description	2204

4.691.2 Member Function Documentation	2204
4.692std::enable_if< bool, _Tp > Struct Template Reference	2205
4.692.1 Detailed Description	2205
4.693std::enable_shared_from_this< _Tp > Class Template Reference	2205
4.693.1 Detailed Description	2206
4.694std::equal_to< _Tp > Struct Template Reference	2206
4.694.1 Detailed Description	2207
4.694.2 Member Typedef Documentation	2207
4.695std::error_category Class Reference	2207
4.695.1 Detailed Description	2208
4.696std::error_code Struct Reference	2208
4.696.1 Detailed Description	2208
4.697std::error_condition Struct Reference	2208
4.697.1 Detailed Description	2209
4.698std::exception Class Reference	2210
4.698.1 Detailed Description	2210
4.698.2 Member Function Documentation	2211
4.699std::exponential_distribution< _RealType > Class Template Reference	2211
4.699.1 Detailed Description	2212
4.699.2 Member Typedef Documentation	2212
4.699.3 Constructor & Destructor Documentation	2212
4.699.4 Member Function Documentation	2212
4.699.5 Friends And Related Function Documentation	2213
4.700std::exponential_distribution< _RealType >::param_type Struct Reference	2214
4.700.1 Detailed Description	2214
4.701std::extent< typename, _UInt > Struct Template Reference	2215
4.701.1 Detailed Description	2215
4.702std::extreme_value_distribution< _RealType > Class Template Reference	2216
4.702.1 Detailed Description	2216
4.702.2 Member Typedef Documentation	2217
4.702.3 Member Function Documentation	2217
4.702.4 Friends And Related Function Documentation	2218
4.703std::extreme_value_distribution< _RealType >::param_type Struct Reference	2218
4.703.1 Detailed Description	2219
4.704std::fisher_f_distribution< _RealType > Class Template Reference	2219
4.704.1 Detailed Description	2220
4.704.2 Member Typedef Documentation	2220

4.704.3 Member Function Documentation	2220
4.704.4 Friends And Related Function Documentation	2221
4.705std::fisher_f_distribution<_RealType>::param_type Struct Reference	2222
4.705.1 Detailed Description	2222
4.706std::forward_iterator_tag Struct Reference	2223
4.706.1 Detailed Description	2223
4.707std::forward_list<_Tp, _Alloc> Class Template Reference	2224
4.707.1 Detailed Description	2226
4.707.2 Constructor & Destructor Documentation	2227
4.707.3 Member Function Documentation	2229
4.708std::fpos<_StateT> Class Template Reference	2240
4.708.1 Detailed Description	2240
4.708.2 Constructor & Destructor Documentation	2241
4.708.3 Member Function Documentation	2241
4.709std::front_insert_iterator<_Container> Class Template Reference	2242
4.709.1 Detailed Description	2243
4.709.2 Member Typedef Documentation	2243
4.709.3 Constructor & Destructor Documentation	2244
4.709.4 Member Function Documentation	2244
4.710std::function<_Res(_ArgTypes...)> Class Template Reference	2245
4.710.1 Detailed Description	2246
4.710.2 Constructor & Destructor Documentation	2246
4.710.3 Member Function Documentation	2248
4.711std::future_error Class Reference	2251
4.711.1 Detailed Description	2251
4.711.2 Member Function Documentation	2251
4.712std::gamma_distribution<_RealType> Class Template Reference	2252
4.712.1 Detailed Description	2253
4.712.2 Member Typedef Documentation	2253
4.712.3 Constructor & Destructor Documentation	2253
4.712.4 Member Function Documentation	2253
4.712.5 Friends And Related Function Documentation	2255
4.713std::gamma_distribution<_RealType>::param_type Struct Reference	2256
4.713.1 Detailed Description	2256
4.714std::geometric_distribution<_IntType> Class Template Reference	2256
4.714.1 Detailed Description	2257
4.714.2 Member Typedef Documentation	2257

4.714.3 Member Function Documentation	2257
4.714.4 Friends And Related Function Documentation	2258
4.715std::geometric_distribution< _IntType >::param_type Struct Reference	2259
4.715.1 Detailed Description	2259
4.716std::greater< _Tp > Struct Template Reference	2260
4.716.1 Detailed Description	2260
4.716.2 Member Typedef Documentation	2260
4.717std::greater_equal< _Tp > Struct Template Reference	2261
4.717.1 Detailed Description	2262
4.717.2 Member Typedef Documentation	2262
4.718std::gslice Class Reference	2262
4.718.1 Detailed Description	2262
4.719std::gslice_array< _Tp > Class Template Reference	2263
4.719.1 Detailed Description	2264
4.720std::has_trivial_copy_assign< _Tp > Struct Template Reference	2265
4.720.1 Detailed Description	2265
4.721std::has_trivial_copy_constructor< _Tp > Struct Template Reference	2266
4.721.1 Detailed Description	2266
4.722std::has_trivial_default_constructor< _Tp > Struct Template Reference	2267
4.722.1 Detailed Description	2268
4.723std::has_virtual_destructor< _Tp > Struct Template Reference	2268
4.723.1 Detailed Description	2269
4.724std::hash< __debug::bitset< _Nb > > Struct Template Reference	2269
4.724.1 Detailed Description	2269
4.725std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference	2269
4.725.1 Detailed Description	2270
4.726std::hash< __gnu_cxx::__u16vstring > Struct Template Reference	2270
4.726.1 Detailed Description	2270
4.727std::hash< __gnu_cxx::__u32vstring > Struct Template Reference	2270
4.727.1 Detailed Description	2271
4.728std::hash< __gnu_cxx::__vstring > Struct Template Reference	2271
4.728.1 Detailed Description	2271
4.729std::hash< __gnu_cxx::__wvstring > Struct Template Reference	2271
4.729.1 Detailed Description	2272
4.730std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference	2272
4.730.1 Detailed Description	2273
4.730.2 Member Typedef Documentation	2273

4.731std::hash< __gnu_cxx::throw_value_random > Struct Template Reference	2274
4.731.1 Detailed Description	2274
4.731.2 Member Typedef Documentation	2274
4.732std::hash< __profile::bitset< _Nb > > Struct Template Reference	2275
4.732.1 Detailed Description	2275
4.733std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference	2275
4.733.1 Detailed Description	2276
4.734std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference	2276
4.734.1 Detailed Description	2276
4.735std::hash< _Tp * > Struct Template Reference	2276
4.735.1 Detailed Description	2277
4.736std::hash< bool > Struct Template Reference	2277
4.736.1 Detailed Description	2277
4.737std::hash< char > Struct Template Reference	2277
4.737.1 Detailed Description	2278
4.738std::hash< char16_t > Struct Template Reference	2278
4.738.1 Detailed Description	2278
4.739std::hash< char32_t > Struct Template Reference	2278
4.739.1 Detailed Description	2279
4.740std::hash< double > Struct Template Reference	2279
4.740.1 Detailed Description	2279
4.741std::hash< error_code > Struct Template Reference	2279
4.741.1 Detailed Description	2280
4.742std::hash< float > Struct Template Reference	2280
4.742.1 Detailed Description	2280
4.743std::hash< int > Struct Template Reference	2280
4.743.1 Detailed Description	2281
4.744std::hash< long > Struct Template Reference	2281
4.744.1 Detailed Description	2281
4.745std::hash< long double > Struct Template Reference	2281
4.745.1 Detailed Description	2282
4.746std::hash< long long > Struct Template Reference	2282
4.746.1 Detailed Description	2282
4.747std::hash< shared_ptr< _Tp > > Struct Template Reference	2282
4.747.1 Detailed Description	2283
4.748std::hash< short > Struct Template Reference	2283
4.748.1 Detailed Description	2283

4.749	std::hash< signed char > Struct Template Reference	2283
4.749.1	Detailed Description	2284
4.750	std::hash< string > Struct Template Reference	2284
4.750.1	Detailed Description	2284
4.751	std::hash< thread::id > Struct Template Reference	2284
4.751.1	Detailed Description	2285
4.752	std::hash< type_index > Struct Template Reference	2285
4.752.1	Detailed Description	2285
4.753	std::hash< u16string > Struct Template Reference	2285
4.753.1	Detailed Description	2286
4.754	std::hash< u32string > Struct Template Reference	2286
4.754.1	Detailed Description	2286
4.755	std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference	2286
4.755.1	Detailed Description	2287
4.756	std::hash< unsigned char > Struct Template Reference	2287
4.756.1	Detailed Description	2287
4.757	std::hash< unsigned int > Struct Template Reference	2287
4.757.1	Detailed Description	2288
4.758	std::hash< unsigned long > Struct Template Reference	2288
4.758.1	Detailed Description	2288
4.759	std::hash< unsigned long long > Struct Template Reference	2288
4.759.1	Detailed Description	2289
4.760	std::hash< unsigned short > Struct Template Reference	2289
4.760.1	Detailed Description	2289
4.761	std::hash< wchar_t > Struct Template Reference	2289
4.761.1	Detailed Description	2290
4.762	std::hash< wstring > Struct Template Reference	2290
4.762.1	Detailed Description	2290
4.763	std::hash<::bitset< _Nb > > Struct Template Reference	2290
4.763.1	Detailed Description	2291
4.764	std::hash<::vector< bool, _Alloc > > Struct Template Reference	2291
4.764.1	Detailed Description	2291
4.765	std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference	2291
4.765.1	Detailed Description	2292
4.765.2	Member Typedef Documentation	2292
4.765.3	Constructor & Destructor Documentation	2292
4.765.4	Member Function Documentation	2293

4.765.5 Friends And Related Function Documentation	2295
4.766std::indirect_array< _Tp > Class Template Reference	2295
4.766.1 Detailed Description	2296
4.767std::initializer_list< _E > Class Template Reference	2297
4.767.1 Detailed Description	2297
4.768std::input_iterator_tag Struct Reference	2298
4.768.1 Detailed Description	2298
4.769std::insert_iterator< _Container > Class Template Reference	2299
4.769.1 Detailed Description	2300
4.769.2 Member Typedef Documentation	2300
4.769.3 Constructor & Destructor Documentation	2300
4.769.4 Member Function Documentation	2301
4.770std::integral_constant< _Tp, __v > Struct Template Reference	2302
4.770.1 Detailed Description	2303
4.771std::invalid_argument Class Reference	2303
4.771.1 Detailed Description	2304
4.771.2 Member Function Documentation	2304
4.772std::ios_base Class Reference	2304
4.772.1 Detailed Description	2306
4.772.2 Member Typedef Documentation	2307
4.772.3 Member Enumeration Documentation	2308
4.772.4 Constructor & Destructor Documentation	2309
4.772.5 Member Function Documentation	2309
4.772.6 Member Data Documentation	2313
4.773std::ios_base::failure Class Reference	2319
4.773.1 Detailed Description	2319
4.773.2 Member Function Documentation	2319
4.774std::is_abstract< _Tp > Struct Template Reference	2320
4.774.1 Detailed Description	2320
4.775std::is_arithmetic< _Tp > Struct Template Reference	2321
4.775.1 Detailed Description	2321
4.776std::is_array< typename > Struct Template Reference	2321
4.776.1 Detailed Description	2322
4.777std::is_assignable< _Tp, _Up > Struct Template Reference	2322
4.777.1 Detailed Description	2323
4.778std::is_base_of< _Base, _Derived > Struct Template Reference	2323
4.778.1 Detailed Description	2324

4.779	<code>std::is_bind_expression< _Tp ></code> Struct Template Reference	2324
4.779.1	Detailed Description	2325
4.780	<code>std::is_bind_expression< _Bind< _Signature > ></code> Struct Template Reference	2325
4.780.1	Detailed Description	2326
4.781	<code>std::is_bind_expression< _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2326
4.781.1	Detailed Description	2326
4.782	<code>std::is_bind_expression< const _Bind< _Signature > ></code> Struct Template Reference	2327
4.782.1	Detailed Description	2327
4.783	<code>std::is_bind_expression< const _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2328
4.783.1	Detailed Description	2328
4.784	<code>std::is_bind_expression< const volatile _Bind< _Signature > ></code> Struct Template Reference	2329
4.784.1	Detailed Description	2329
4.785	<code>std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2330
4.785.1	Detailed Description	2330
4.786	<code>std::is_bind_expression< volatile _Bind< _Signature > ></code> Struct Template Reference	2331
4.786.1	Detailed Description	2331
4.787	<code>std::is_bind_expression< volatile _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2332
4.787.1	Detailed Description	2332
4.788	<code>std::is_class< _Tp ></code> Struct Template Reference	2333
4.788.1	Detailed Description	2333
4.789	<code>std::is_compound< _Tp ></code> Struct Template Reference	2334
4.789.1	Detailed Description	2334
4.790	<code>std::is_const< typename ></code> Struct Template Reference	2335
4.790.1	Detailed Description	2335
4.791	<code>std::is_constructible< _Tp, _Args ></code> Struct Template Reference	2336
4.791.1	Detailed Description	2336
4.792	<code>std::is_convertible< _From, _To ></code> Struct Template Reference	2337
4.792.1	Detailed Description	2338
4.793	<code>std::is_copy_assignable< _Tp ></code> Struct Template Reference	2338
4.793.1	Detailed Description	2338
4.794	<code>std::is_copy_constructible< _Tp ></code> Struct Template Reference	2338
4.794.1	Detailed Description	2338
4.795	<code>std::is_default_constructible< _Tp ></code> Struct Template Reference	2339
4.795.1	Detailed Description	2339
4.796	<code>std::is_destructible< _Tp ></code> Struct Template Reference	2340
4.796.1	Detailed Description	2340

4.797std::is_empty< _Tp > Struct Template Reference	2341
4.797.1 Detailed Description	2341
4.798std::is_enum< _Tp > Struct Template Reference	2342
4.798.1 Detailed Description	2343
4.799std::is_error_code_enum< _Tp > Struct Template Reference	2343
4.799.1 Detailed Description	2343
4.800std::is_error_code_enum< future_errc > Struct Template Reference	2344
4.800.1 Detailed Description	2344
4.801std::is_error_condition_enum< _Tp > Struct Template Reference	2345
4.801.1 Detailed Description	2345
4.802std::is_floating_point< _Tp > Struct Template Reference	2346
4.802.1 Detailed Description	2346
4.803std::is_function< typename > Struct Template Reference	2347
4.803.1 Detailed Description	2347
4.804std::is_fundamental< _Tp > Struct Template Reference	2347
4.804.1 Detailed Description	2348
4.805std::is_integral< _Tp > Struct Template Reference	2348
4.805.1 Detailed Description	2348
4.806std::is_literal_type< _Tp > Struct Template Reference	2349
4.806.1 Detailed Description	2349
4.807std::is_lvalue_reference< typename > Struct Template Reference	2350
4.807.1 Detailed Description	2350
4.808std::is_member_function_pointer< _Tp > Struct Template Reference	2351
4.808.1 Detailed Description	2351
4.809std::is_member_object_pointer< _Tp > Struct Template Reference	2351
4.809.1 Detailed Description	2352
4.810std::is_member_pointer< _Tp > Struct Template Reference	2352
4.810.1 Detailed Description	2353
4.811std::is_move_assignable< _Tp > Struct Template Reference	2353
4.811.1 Detailed Description	2353
4.812std::is_move_constructible< _Tp > Struct Template Reference	2353
4.812.1 Detailed Description	2353
4.813std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference	2354
4.813.1 Detailed Description	2354
4.814std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference	2354
4.814.1 Detailed Description	2354
4.815std::is_nothrow_copy_assignable< _Tp > Struct Template Reference	2355

4.815	std::is_nothrow_copy_constructible< _Tp > Struct Template Reference	2355
4.815.1	Detailed Description	2355
4.816	std::is_nothrow_default_constructible< _Tp > Struct Template Reference	2355
4.816.1	Detailed Description	2355
4.817	std::is_nothrow_destructible< _Tp > Struct Template Reference	2356
4.817.1	Detailed Description	2356
4.818	std::is_nothrow_move_assignable< _Tp > Struct Template Reference	2357
4.818.1	Detailed Description	2357
4.819	std::is_nothrow_move_constructible< _Tp > Struct Template Reference	2357
4.819.1	Detailed Description	2357
4.820	std::is_object< _Tp > Struct Template Reference	2358
4.820.1	Detailed Description	2358
4.821	std::is_placeholder< _Tp > Struct Template Reference	2359
4.821.1	Detailed Description	2359
4.822	std::is_placeholder< _Placeholder< _Num > > Struct Template Reference	2360
4.822.1	Detailed Description	2360
4.823	std::is_pod< _Tp > Struct Template Reference	2361
4.823.1	Detailed Description	2361
4.824	std::is_pointer< _Tp > Struct Template Reference	2362
4.824.1	Detailed Description	2362
4.825	std::is_polymorphic< _Tp > Struct Template Reference	2363
4.825.1	Detailed Description	2363
4.826	std::is_reference< _Tp > Struct Template Reference	2364
4.826.1	Detailed Description	2364
4.827	std::is_rvalue_reference< typename > Struct Template Reference	2364
4.827.1	Detailed Description	2365
4.828	std::is_same< typename, typename > Struct Template Reference	2365
4.828.1	Detailed Description	2365
4.829	std::is_scalar< _Tp > Struct Template Reference	2366
4.829.1	Detailed Description	2366
4.830	std::is_signed< _Tp > Struct Template Reference	2366
4.830.1	Detailed Description	2366
4.831	std::is_standard_layout< _Tp > Struct Template Reference	2368
4.831.1	Detailed Description	2368
4.832	std::is_trivial< _Tp > Struct Template Reference	2369
4.832.1	Detailed Description	2369

4.834std::is_trivially_destructible< _Tp > Struct Template Reference	2370
4.834.1 Detailed Description	2370
4.835std::is_union< _Tp > Struct Template Reference	2370
4.835.1 Detailed Description	2371
4.836std::is_unsigned< _Tp > Struct Template Reference	2371
4.836.1 Detailed Description	2371
4.837std::is_void< _Tp > Struct Template Reference	2371
4.837.1 Detailed Description	2372
4.838std::is_volatile< typename > Struct Template Reference	2372
4.838.1 Detailed Description	2373
4.839std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference	2373
4.839.1 Detailed Description	2374
4.839.2 Member Typedef Documentation	2374
4.839.3 Constructor & Destructor Documentation	2375
4.840std::istreambuf_iterator< _CharT, _Traits > Class Template Reference	2375
4.840.1 Detailed Description	2377
4.840.2 Member Typedef Documentation	2377
4.840.3 Constructor & Destructor Documentation	2378
4.840.4 Member Function Documentation	2378
4.841std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference	2380
4.841.1 Detailed Description	2380
4.841.2 Member Typedef Documentation	2381
4.842std::iterator_traits< _Tp * > Struct Template Reference	2381
4.842.1 Detailed Description	2382
4.843std::iterator_traits< const _Tp * > Struct Template Reference	2382
4.843.1 Detailed Description	2382
4.844std::length_error Class Reference	2383
4.844.1 Detailed Description	2383
4.844.2 Member Function Documentation	2383
4.845std::less< _Tp > Struct Template Reference	2384
4.845.1 Detailed Description	2384
4.845.2 Member Typedef Documentation	2384
4.846std::less_equal< _Tp > Struct Template Reference	2385
4.846.1 Detailed Description	2386
4.846.2 Member Typedef Documentation	2386
4.847std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	2386
4.847.1 Detailed Description	2387

4.847.2 Member Typedef Documentation	2387
4.847.3 Constructor & Destructor Documentation	2387
4.847.4 Member Function Documentation	2388
4.847.5 Friends And Related Function Documentation	2389
4.847.6 Member Data Documentation	2390
4.848std::list< _Tp, _Alloc > Class Template Reference	2391
4.848.1 Detailed Description	2394
4.848.2 Constructor & Destructor Documentation	2394
4.848.3 Member Function Documentation	2396
4.849std::locale Class Reference	2408
4.849.1 Detailed Description	2410
4.849.2 Member Typedef Documentation	2410
4.849.3 Constructor & Destructor Documentation	2410
4.849.4 Member Function Documentation	2412
4.849.5 Friends And Related Function Documentation	2414
4.849.6 Member Data Documentation	2415
4.850std::locale::facet Class Reference	2417
4.850.1 Detailed Description	2418
4.850.2 Constructor & Destructor Documentation	2418
4.851std::locale::id Class Reference	2418
4.851.1 Detailed Description	2419
4.851.2 Constructor & Destructor Documentation	2419
4.851.3 Friends And Related Function Documentation	2419
4.852std::lock_guard< _Mutex > Class Template Reference	2420
4.852.1 Detailed Description	2420
4.853std::logic_error Class Reference	2421
4.853.1 Detailed Description	2421
4.853.2 Constructor & Destructor Documentation	2421
4.853.3 Member Function Documentation	2421
4.854std::logical_and< _Tp > Struct Template Reference	2422
4.854.1 Detailed Description	2422
4.854.2 Member Typedef Documentation	2423
4.855std::logical_not< _Tp > Struct Template Reference	2423
4.855.1 Detailed Description	2424
4.855.2 Member Typedef Documentation	2424
4.856std::logical_or< _Tp > Struct Template Reference	2425
4.856.1 Detailed Description	2425

4.856.2 Member Typedef Documentation	2425
4.857std::lognormal_distribution< _RealType > Class Template Reference	2426
4.857.1 Detailed Description	2427
4.857.2 Member Typedef Documentation	2427
4.857.3 Member Function Documentation	2427
4.857.4 Friends And Related Function Documentation	2428
4.858std::lognormal_distribution< _RealType >::param_type Struct Reference	2429
4.858.1 Detailed Description	2430
4.859std::make_signed< _Tp > Struct Template Reference	2430
4.859.1 Detailed Description	2430
4.860std::make_unsigned< _Tp > Struct Template Reference	2430
4.860.1 Detailed Description	2430
4.861std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2431
4.861.1 Detailed Description	2433
4.861.2 Constructor & Destructor Documentation	2433
4.861.3 Member Function Documentation	2435
4.862std::mask_array< _Tp > Class Template Reference	2446
4.862.1 Detailed Description	2447
4.863std::match_results< _Bi_iter, _Alloc > Class Template Reference	2448
4.863.1 Detailed Description	2452
4.863.2 Constructor & Destructor Documentation	2452
4.863.3 Member Function Documentation	2453
4.864std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2458
4.864.1 Detailed Description	2459
4.864.2 Member Typedef Documentation	2459
4.865std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2460
4.865.1 Detailed Description	2460
4.865.2 Member Typedef Documentation	2460
4.866std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2461
4.866.1 Detailed Description	2462
4.866.2 Member Typedef Documentation	2462
4.867std::mem_fun_t< _Ret, _Tp > Class Template Reference	2462
4.867.1 Detailed Description	2463
4.867.2 Member Typedef Documentation	2463
4.868std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference	2463
4.868.1 Detailed Description	2465

4.868.2 Member Typedef Documentation	2465
4.868.3 Constructor & Destructor Documentation	2465
4.868.4 Member Function Documentation	2466
4.868.5 Friends And Related Function Documentation	2466
4.869std::messages< _CharT > Class Template Reference	2468
4.869.1 Detailed Description	2469
4.869.2 Member Typedef Documentation	2469
4.869.3 Constructor & Destructor Documentation	2470
4.869.4 Member Data Documentation	2470
4.870std::messages_base Struct Reference	2471
4.870.1 Detailed Description	2471
4.871std::messages_byname< _CharT > Class Template Reference	2472
4.871.1 Detailed Description	2473
4.871.2 Member Data Documentation	2473
4.872std::minus< _Tp > Struct Template Reference	2474
4.872.1 Detailed Description	2474
4.872.2 Member Typedef Documentation	2474
4.873std::modulus< _Tp > Struct Template Reference	2475
4.873.1 Detailed Description	2476
4.873.2 Member Typedef Documentation	2476
4.874std::money_base Class Reference	2476
4.874.1 Detailed Description	2477
4.875std::money_get< _CharT, _InIter > Class Template Reference	2477
4.875.1 Detailed Description	2478
4.875.2 Member Typedef Documentation	2479
4.875.3 Constructor & Destructor Documentation	2479
4.875.4 Member Function Documentation	2479
4.875.5 Member Data Documentation	2481
4.876std::money_put< _CharT, _OutIter > Class Template Reference	2482
4.876.1 Detailed Description	2483
4.876.2 Member Typedef Documentation	2483
4.876.3 Constructor & Destructor Documentation	2483
4.876.4 Member Function Documentation	2484
4.876.5 Member Data Documentation	2486
4.877std::money_punct< _CharT, _Intl > Class Template Reference	2486
4.877.1 Detailed Description	2488
4.877.2 Member Typedef Documentation	2488

4.877.3 Constructor & Destructor Documentation	2488
4.877.4 Member Function Documentation	2489
4.877.5 Member Data Documentation	2495
4.878std::moneypunct_byname< _CharT, _Intl > Class Template Reference	2495
4.878.1 Detailed Description	2497
4.878.2 Member Function Documentation	2497
4.878.3 Member Data Documentation	2503
4.879std::move_iterator< _Iterator > Class Template Reference	2503
4.879.1 Detailed Description	2504
4.880std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2504
4.880.1 Detailed Description	2506
4.880.2 Constructor & Destructor Documentation	2507
4.880.3 Member Function Documentation	2508
4.881std::multiplies< _Tp > Struct Template Reference	2519
4.881.1 Detailed Description	2520
4.881.2 Member Typedef Documentation	2520
4.882std::multiset< _Key, _Compare, _Alloc > Class Template Reference	2520
4.882.1 Detailed Description	2522
4.882.2 Constructor & Destructor Documentation	2523
4.882.3 Member Function Documentation	2524
4.883std::mutex Class Reference	2534
4.883.1 Detailed Description	2534
4.884std::negate< _Tp > Struct Template Reference	2535
4.884.1 Detailed Description	2535
4.884.2 Member Typedef Documentation	2535
4.885std::negative_binomial_distribution< _IntType > Class Template Reference	2536
4.885.1 Detailed Description	2537
4.885.2 Member Typedef Documentation	2537
4.885.3 Member Function Documentation	2537
4.885.4 Friends And Related Function Documentation	2538
4.886std::negative_binomial_distribution< _IntType >::param_type Struct Reference	2539
4.886.1 Detailed Description	2540
4.887std::nested_exception Class Reference	2540
4.887.1 Detailed Description	2540
4.888std::normal_distribution< _RealType > Class Template Reference	2540
4.888.1 Detailed Description	2542
4.888.2 Member Typedef Documentation	2542

4.888.3 Constructor & Destructor Documentation	2542
4.888.4 Member Function Documentation	2542
4.888.5 Friends And Related Function Documentation	2543
4.889std::normal_distribution< _RealType >::param_type Struct Reference	2544
4.889.1 Detailed Description	2545
4.890std::not_equal_to< _Tp > Struct Template Reference	2545
4.890.1 Detailed Description	2546
4.890.2 Member Typedef Documentation	2546
4.891std::num_get< _CharT, _InIter > Class Template Reference	2546
4.891.1 Detailed Description	2548
4.891.2 Member Typedef Documentation	2548
4.891.3 Constructor & Destructor Documentation	2548
4.891.4 Member Function Documentation	2549
4.891.5 Member Data Documentation	2560
4.892std::num_put< _CharT, _OutIter > Class Template Reference	2560
4.892.1 Detailed Description	2562
4.892.2 Member Typedef Documentation	2562
4.892.3 Constructor & Destructor Documentation	2562
4.892.4 Member Function Documentation	2563
4.892.5 Member Data Documentation	2570
4.893std::numeric_limits< _Tp > Struct Template Reference	2571
4.893.1 Detailed Description	2572
4.893.2 Member Function Documentation	2572
4.893.3 Member Data Documentation	2573
4.894std::numeric_limits< bool > Struct Template Reference	2576
4.894.1 Detailed Description	2577
4.895std::numeric_limits< char > Struct Template Reference	2577
4.895.1 Detailed Description	2578
4.896std::numeric_limits< char16_t > Struct Template Reference	2578
4.896.1 Detailed Description	2579
4.897std::numeric_limits< char32_t > Struct Template Reference	2579
4.897.1 Detailed Description	2580
4.898std::numeric_limits< double > Struct Template Reference	2580
4.898.1 Detailed Description	2581
4.899std::numeric_limits< float > Struct Template Reference	2581
4.899.1 Detailed Description	2582
4.900std::numeric_limits< int > Struct Template Reference	2582

4.900.1 Detailed Description	2583
4.901std::numeric_limits< long > Struct Template Reference	2583
4.901.1 Detailed Description	2584
4.902std::numeric_limits< long double > Struct Template Reference	2584
4.902.1 Detailed Description	2585
4.903std::numeric_limits< long long > Struct Template Reference	2585
4.903.1 Detailed Description	2586
4.904std::numeric_limits< short > Struct Template Reference	2586
4.904.1 Detailed Description	2587
4.905std::numeric_limits< signed char > Struct Template Reference	2587
4.905.1 Detailed Description	2588
4.906std::numeric_limits< unsigned char > Struct Template Reference	2588
4.906.1 Detailed Description	2589
4.907std::numeric_limits< unsigned int > Struct Template Reference	2589
4.907.1 Detailed Description	2590
4.908std::numeric_limits< unsigned long > Struct Template Reference	2590
4.908.1 Detailed Description	2591
4.909std::numeric_limits< unsigned long long > Struct Template Reference	2591
4.909.1 Detailed Description	2592
4.910std::numeric_limits< unsigned short > Struct Template Reference	2592
4.910.1 Detailed Description	2593
4.911std::numeric_limits< wchar_t > Struct Template Reference	2593
4.911.1 Detailed Description	2594
4.912std::num_punct< _CharT > Class Template Reference	2595
4.912.1 Detailed Description	2596
4.912.2 Member Typedef Documentation	2596
4.912.3 Constructor & Destructor Documentation	2597
4.912.4 Member Function Documentation	2597
4.912.5 Member Data Documentation	2600
4.913std::num_punct_byname< _CharT > Class Template Reference	2601
4.913.1 Detailed Description	2602
4.913.2 Member Function Documentation	2602
4.913.3 Member Data Documentation	2605
4.914std::once_flag Struct Reference	2605
4.914.1 Detailed Description	2605
4.914.2 Constructor & Destructor Documentation	2605
4.914.3 Member Function Documentation	2605

4.914.4 Friends And Related Function Documentation	2606
4.915std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	2606
4.915.1 Detailed Description	2607
4.915.2 Member Typedef Documentation	2607
4.915.3 Constructor & Destructor Documentation	2608
4.915.4 Member Function Documentation	2609
4.916std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	2609
4.916.1 Detailed Description	2610
4.916.2 Member Typedef Documentation	2610
4.916.3 Constructor & Destructor Documentation	2611
4.916.4 Member Function Documentation	2612
4.917std::out_of_range Class Reference	2613
4.917.1 Detailed Description	2613
4.917.2 Member Function Documentation	2613
4.918std::output_iterator_tag Struct Reference	2613
4.918.1 Detailed Description	2613
4.919std::overflow_error Class Reference	2614
4.919.1 Detailed Description	2614
4.919.2 Member Function Documentation	2614
4.920std::owner_less< shared_ptr< _Tp > > Struct Template Reference	2615
4.920.1 Detailed Description	2615
4.920.2 Member Typedef Documentation	2615
4.921std::owner_less< weak_ptr< _Tp > > Struct Template Reference	2615
4.921.1 Detailed Description	2616
4.921.2 Member Typedef Documentation	2616
4.922std::pair< _T1, _T2 > Struct Template Reference	2617
4.922.1 Detailed Description	2618
4.922.2 Member Typedef Documentation	2618
4.922.3 Constructor & Destructor Documentation	2618
4.922.4 Member Data Documentation	2619
4.923std::piecewise_constant_distribution< _RealType > Class Template Reference	2619
4.923.1 Detailed Description	2620
4.923.2 Member Typedef Documentation	2620
4.923.3 Member Function Documentation	2620
4.923.4 Friends And Related Function Documentation	2622
4.924std::piecewise_constant_distribution< _RealType >::param_type Struct Reference	2622
4.924.1 Detailed Description	2623

4.925std::piecewise_construct_t Struct Reference	2623
4.925.1 Detailed Description	2623
4.926std::piecewise_linear_distribution<_RealType> Class Template Reference	2623
4.926.1 Detailed Description	2624
4.926.2 Member Typedef Documentation	2625
4.926.3 Member Function Documentation	2625
4.926.4 Friends And Related Function Documentation	2626
4.927std::piecewise_linear_distribution<_RealType>::param_type Struct Reference	2627
4.927.1 Detailed Description	2627
4.928std::plus<_Tp> Struct Template Reference	2628
4.928.1 Detailed Description	2628
4.928.2 Member Typedef Documentation	2628
4.929std::pointer_to_binary_function<_Arg1, _Arg2, _Result> Class Template Reference	2629
4.929.1 Detailed Description	2630
4.929.2 Member Typedef Documentation	2630
4.930std::pointer_to_unary_function<_Arg, _Result> Class Template Reference	2630
4.930.1 Detailed Description	2631
4.930.2 Member Typedef Documentation	2631
4.931std::pointer_traits<_Ptr> Struct Template Reference	2631
4.931.1 Detailed Description	2632
4.931.2 Member Typedef Documentation	2632
4.932std::pointer_traits<_Tp*> Struct Template Reference	2632
4.932.1 Detailed Description	2633
4.932.2 Member Typedef Documentation	2633
4.932.3 Member Function Documentation	2633
4.933std::poisson_distribution<_IntType> Class Template Reference	2633
4.933.1 Detailed Description	2634
4.933.2 Member Typedef Documentation	2635
4.933.3 Member Function Documentation	2635
4.933.4 Friends And Related Function Documentation	2636
4.934std::poisson_distribution<_IntType>::param_type Struct Reference	2637
4.934.1 Detailed Description	2637
4.935std::priority_queue<_Tp, _Sequence, _Compare> Class Template Reference	2637
4.935.1 Detailed Description	2638
4.935.2 Constructor & Destructor Documentation	2639
4.935.3 Member Function Documentation	2639
4.936std::queue<_Tp, _Sequence> Class Template Reference	2640

4.936.1 Detailed Description	2641
4.936.2 Constructor & Destructor Documentation	2642
4.936.3 Member Function Documentation	2642
4.936.4 Member Data Documentation	2643
4.937std::random_access_iterator_tag Struct Reference	2644
4.937.1 Detailed Description	2644
4.938std::random_device Class Reference	2644
4.938.1 Detailed Description	2645
4.938.2 Member Typedef Documentation	2645
4.939std::range_error Class Reference	2645
4.939.1 Detailed Description	2646
4.939.2 Member Function Documentation	2646
4.940std::rank< typename > Struct Template Reference	2646
4.940.1 Detailed Description	2647
4.941std::ratio< _Num, _Den > Struct Template Reference	2647
4.941.1 Detailed Description	2647
4.942std::ratio_equal< _R1, _R2 > Struct Template Reference	2648
4.942.1 Detailed Description	2648
4.943std::ratio_not_equal< _R1, _R2 > Struct Template Reference	2649
4.943.1 Detailed Description	2649
4.944std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	2650
4.944.1 Detailed Description	2651
4.944.2 Member Typedef Documentation	2651
4.945std::recursive_mutex Class Reference	2651
4.945.1 Detailed Description	2652
4.946std::reference_wrapper< _Tp > Class Template Reference	2652
4.946.1 Detailed Description	2653
4.947std::regex_error Class Reference	2653
4.947.1 Detailed Description	2653
4.947.2 Constructor & Destructor Documentation	2654
4.947.3 Member Function Documentation	2654
4.948std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	2654
4.948.1 Detailed Description	2655
4.948.2 Constructor & Destructor Documentation	2655
4.948.3 Member Function Documentation	2656
4.949std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	2657
4.949.1 Detailed Description	2657

4.949.2 Constructor & Destructor Documentation	2658
4.949.3 Member Function Documentation	2659
4.950std::regex_traits< _Ch_type > Struct Template Reference	2660
4.950.1 Detailed Description	2661
4.950.2 Constructor & Destructor Documentation	2662
4.950.3 Member Function Documentation	2662
4.951std::remove_all_extents< _Tp > Struct Template Reference	2665
4.951.1 Detailed Description	2665
4.952std::remove_const< _Tp > Struct Template Reference	2666
4.952.1 Detailed Description	2666
4.953std::remove_cv< _Tp > Struct Template Reference	2666
4.953.1 Detailed Description	2666
4.954std::remove_extent< _Tp > Struct Template Reference	2667
4.954.1 Detailed Description	2667
4.955std::remove_pointer< _Tp > Struct Template Reference	2667
4.955.1 Detailed Description	2667
4.956std::remove_reference< _Tp > Struct Template Reference	2667
4.956.1 Detailed Description	2668
4.957std::remove_volatile< _Tp > Struct Template Reference	2668
4.957.1 Detailed Description	2668
4.958std::reverse_iterator< _Iterator > Class Template Reference	2668
4.958.1 Detailed Description	2669
4.958.2 Member Typedef Documentation	2670
4.958.3 Constructor & Destructor Documentation	2670
4.958.4 Member Function Documentation	2670
4.959std::runtime_error Class Reference	2673
4.959.1 Detailed Description	2673
4.959.2 Constructor & Destructor Documentation	2673
4.959.3 Member Function Documentation	2674
4.960std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference	2674
4.960.1 Detailed Description	2676
4.961std::seed_seq Class Reference	2676
4.961.1 Detailed Description	2676
4.961.2 Member Typedef Documentation	2676
4.961.3 Constructor & Destructor Documentation	2676
4.962std::set< _Key, _Compare, _Alloc > Class Template Reference	2677
4.962.1 Detailed Description	2679

4.962.2 Member Typedef Documentation	2679
4.962.3 Constructor & Destructor Documentation	2681
4.962.4 Member Function Documentation	2683
4.963std::shared_ptr< _Tp > Class Template Reference	2692
4.963.1 Detailed Description	2694
4.963.2 Constructor & Destructor Documentation	2694
4.963.3 Friends And Related Function Documentation	2698
4.964std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference	2698
4.964.1 Detailed Description	2699
4.964.2 Member Typedef Documentation	2700
4.964.3 Constructor & Destructor Documentation	2700
4.964.4 Member Function Documentation	2701
4.964.5 Friends And Related Function Documentation	2702
4.965std::slice Class Reference	2703
4.965.1 Detailed Description	2703
4.966std::slice_array< _Tp > Class Template Reference	2704
4.966.1 Detailed Description	2705
4.967std::stack< _Tp, _Sequence > Class Template Reference	2705
4.967.1 Detailed Description	2706
4.967.2 Constructor & Destructor Documentation	2707
4.967.3 Member Function Documentation	2707
4.968std::student_t_distribution< _RealType > Class Template Reference	2708
4.968.1 Detailed Description	2709
4.968.2 Member Typedef Documentation	2709
4.968.3 Member Function Documentation	2709
4.968.4 Friends And Related Function Documentation	2710
4.969std::student_t_distribution< _RealType >::param_type Struct Reference	2711
4.969.1 Detailed Description	2711
4.970std::sub_match< _Bilter > Class Template Reference	2712
4.970.1 Detailed Description	2713
4.970.2 Member Typedef Documentation	2713
4.970.3 Member Function Documentation	2713
4.970.4 Member Data Documentation	2715
4.971std::system_error Class Reference	2715
4.971.1 Detailed Description	2716
4.971.2 Member Function Documentation	2716
4.972std::thread Class Reference	2716

4.972.1 Detailed Description	2717
4.972.2 Member Function Documentation	2717
4.973std::thread::id Class Reference	2717
4.973.1 Detailed Description	2717
4.974std::time_base Class Reference	2718
4.974.1 Detailed Description	2718
4.975std::time_get< _CharT, _InIter > Class Template Reference	2719
4.975.1 Detailed Description	2720
4.975.2 Member Typedef Documentation	2720
4.975.3 Constructor & Destructor Documentation	2721
4.975.4 Member Function Documentation	2721
4.975.5 Member Data Documentation	2727
4.976std::time_get_byname< _CharT, _InIter > Class Template Reference	2727
4.976.1 Detailed Description	2729
4.976.2 Member Function Documentation	2729
4.976.3 Member Data Documentation	2735
4.977std::time_put< _CharT, _OutIter > Class Template Reference	2735
4.977.1 Detailed Description	2736
4.977.2 Member Typedef Documentation	2736
4.977.3 Constructor & Destructor Documentation	2737
4.977.4 Member Function Documentation	2737
4.977.5 Member Data Documentation	2739
4.978std::time_put_byname< _CharT, _OutIter > Class Template Reference	2739
4.978.1 Detailed Description	2740
4.978.2 Member Function Documentation	2740
4.978.3 Member Data Documentation	2742
4.979std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference	2742
4.979.1 Detailed Description	2744
4.979.2 Member Data Documentation	2744
4.980std::tr2::__reflection_typelist< _First, _Rest...> Struct Template Reference	2744
4.980.1 Detailed Description	2744
4.981std::tr2::__reflection_typelist<> Struct Template Reference	2744
4.981.1 Detailed Description	2744
4.982std::tr2::bases< _Tp > Struct Template Reference	2745
4.982.1 Detailed Description	2745
4.983std::tr2::bool_set Class Reference	2745
4.983.1 Detailed Description	2746

4.983.2 Constructor & Destructor Documentation	2746
4.983.3 Member Function Documentation	2746
4.984std::tr2::direct_bases< _Tp > Struct Template Reference	2747
4.984.1 Detailed Description	2747
4.985std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference	2747
4.985.1 Detailed Description	2751
4.985.2 Constructor & Destructor Documentation	2751
4.985.3 Member Function Documentation	2753
4.986std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference	2761
4.986.1 Detailed Description	2761
4.987std::try_to_lock_t Struct Reference	2762
4.987.1 Detailed Description	2762
4.988std::tuple< _Elements > Class Template Reference	2762
4.988.1 Detailed Description	2763
4.989std::tuple< _T1, _T2 > Class Template Reference	2764
4.989.1 Detailed Description	2765
4.990std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference	2765
4.990.1 Detailed Description	2765
4.991std::tuple_element< __i, tuple< _Head, _Tail...> > Struct Template Reference	2766
4.991.1 Detailed Description	2766
4.992std::tuple_size< tuple< _Elements...> > Struct Template Reference	2767
4.992.1 Detailed Description	2767
4.993std::type_index Struct Reference	2768
4.993.1 Detailed Description	2768
4.994std::type_info Class Reference	2768
4.994.1 Detailed Description	2769
4.994.2 Constructor & Destructor Documentation	2769
4.994.3 Member Function Documentation	2769
4.995std::unary_function< _Arg, _Result > Struct Template Reference	2770
4.995.1 Detailed Description	2771
4.995.2 Member Typedef Documentation	2771
4.996std::unary_negate< _Predicate > Class Template Reference	2772
4.996.1 Detailed Description	2772
4.996.2 Member Typedef Documentation	2773
4.997std::underflow_error Class Reference	2773
4.997.1 Detailed Description	2773
4.997.2 Member Function Documentation	2774

4.998	<code>std::underlying_type< _Tp ></code> Struct Template Reference	2774
4.998.1	Detailed Description	2774
4.999	<code>std::uniform_int_distribution< _IntType ></code> Class Template Reference	2774
4.999.1	Detailed Description	2775
4.999.2	Member Typedef Documentation	2775
4.999.3	Constructor & Destructor Documentation	2775
4.999.4	Member Function Documentation	2775
4.999.5	Friends And Related Function Documentation	2776
4.1000	<code>std::uniform_int_distribution< _IntType >::param_type</code> Struct Reference	2777
4.1000.1	Detailed Description	2777
4.1001	<code>std::uniform_real_distribution< _RealType ></code> Class Template Reference	2777
4.1001.1	Detailed Description	2778
4.1001.2	Member Typedef Documentation	2778
4.1001.3	Constructor & Destructor Documentation	2778
4.1001.4	Member Function Documentation	2779
4.1001.5	Friends And Related Function Documentation	2780
4.1002	<code>std::uniform_real_distribution< _RealType >::param_type</code> Struct Reference	2780
4.1002.1	Detailed Description	2780
4.1003	<code>std::unique_lock< _Mutex ></code> Class Template Reference	2780
4.1003.1	Detailed Description	2781
4.1004	<code>std::unique_ptr< _Tp, _Dp ></code> Class Template Reference	2781
4.1004.1	Detailed Description	2782
4.1005	<code>std::unique_ptr< _Tp[], _Dp ></code> Class Template Reference	2783
4.1005.1	Detailed Description	2784
4.1006	<code>std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc ></code> Class Template Reference	2784
4.1006.1	Detailed Description	2786
4.1006.2	Member Typedef Documentation	2786
4.1006.3	Constructor & Destructor Documentation	2789
4.1006.4	Member Function Documentation	2790
4.1007	<code>std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc ></code> Class Template Reference	2804
4.1007.1	Detailed Description	2806
4.1007.2	Member Typedef Documentation	2806
4.1007.3	Constructor & Destructor Documentation	2808
4.1007.4	Member Function Documentation	2810
4.1008	<code>std::unordered_multiset< _Value, _Hash, _Pred, _Alloc ></code> Class Template Reference	2822
4.1008.1	Detailed Description	2824
4.1008.2	Member Typedef Documentation	2825

4.1008.3	Constructor & Destructor Documentation	2827
4.1008.4	Member Function Documentation	2828
4.1009	std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	2840
4.1009.1	Detailed Description	2842
4.1009.2	Member Typedef Documentation	2842
4.1009.3	Constructor & Destructor Documentation	2844
4.1009.4	Member Function Documentation	2846
4.1010	std::uses_allocator< _Tp, _Alloc > Struct Template Reference	2859
4.1010.1	Detailed Description	2859
4.1011	std::uses_allocator< tuple< _Types..., _Alloc > Struct Template Reference	2860
4.1011.1	Detailed Description	2860
4.1012	std::valarray< _Tp > Class Template Reference	2861
4.1012.1	Detailed Description	2863
4.1012.2	Constructor & Destructor Documentation	2863
4.1013	std::vector< _Tp, _Alloc > Class Template Reference	2864
4.1013.1	Detailed Description	2867
4.1013.2	Constructor & Destructor Documentation	2867
4.1013.3	Member Function Documentation	2870
4.1014	std::vector< bool, _Alloc > Class Template Reference	2881
4.1014.1	Detailed Description	2884
4.1015	std::weak_ptr< _Tp > Class Template Reference	2884
4.1015.1	Detailed Description	2885
4.1016	std::weibull_distribution< _RealType > Class Template Reference	2885
4.1016.1	Detailed Description	2886
4.1016.2	Member Typedef Documentation	2886
4.1016.3	Member Function Documentation	2886
4.1016.4	Friends And Related Function Documentation	2887
4.1017	std::weibull_distribution< _RealType >::param_type Struct Reference	2888
4.1017.1	Detailed Description	2888
4.1018	free_metadata_helper< Node_Update, _BTp > Struct Template Reference	2888
4.1018.1	Detailed Description	2888
4.1019	free_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference	2889
4.1019.1	Detailed Description	2889
4.1020	free_metadata_helper< Node_Update, _BTp > Struct Template Reference	2889
4.1020.1	Detailed Description	2889
4.1021	free_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference	2889
4.1021.1	Detailed Description	2889

4.1022	tuple_element< _Int, _Tp > Class Template Reference	2890
4.1022.1	Detailed Description	2890
4.1023	tuple_element< __i, _Tp > Struct Template Reference	2890
4.1023.1	Detailed Description	2890
4.1024	tuple_size< _Tp > Class Template Reference	2890
4.1024.1	Detailed Description	2890
4.1025	tuple_size< _Tp > Struct Template Reference	2891
4.1025.1	Detailed Description	2891
4.1026	type_base< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	2891
4.1026.1	Detailed Description	2891
4.1027	uses_allocator< typename, typename > Struct Template Reference	2891
4.1027.1	Detailed Description	2891
5	File Documentation	2892
5.1	algo.h File Reference	2892
5.1.1	Detailed Description	2901
5.2	algbase.h File Reference	2902
5.2.1	Detailed Description	2903
5.3	algorithm File Reference	2903
5.3.1	Detailed Description	2903
5.4	algorithm File Reference	2903
5.4.1	Detailed Description	2904
5.5	algorithm File Reference	2904
5.5.1	Detailed Description	2904
5.6	algorithmfwd.h File Reference	2904
5.6.1	Detailed Description	2910
5.7	algorithmfwd.h File Reference	2910
5.7.1	Detailed Description	2919
5.8	alloc_traits.h File Reference	2919
5.8.1	Detailed Description	2919
5.9	alloc_traits.h File Reference	2920
5.9.1	Detailed Description	2920
5.10	allocator.h File Reference	2920
5.10.1	Detailed Description	2921
5.11	array File Reference	2921
5.11.1	Detailed Description	2922
5.12	array_allocator.h File Reference	2922

5.12.1 Detailed Description	2922
5.13 assoc_container.hpp File Reference	2922
5.13.1 Detailed Description	2923
5.14 atomic File Reference	2923
5.14.1 Detailed Description	2926
5.15 atomic_base.h File Reference	2927
5.15.1 Detailed Description	2929
5.16 atomic_lockfree_defines.h File Reference	2929
5.16.1 Detailed Description	2929
5.17 atomic_word.h File Reference	2929
5.17.1 Detailed Description	2929
5.18 atomicity.h File Reference	2929
5.18.1 Detailed Description	2930
5.19 auto_ptr.h File Reference	2930
5.19.1 Detailed Description	2930
5.20 backward_warning.h File Reference	2931
5.20.1 Detailed Description	2931
5.21 balanced_quicksort.h File Reference	2931
5.21.1 Detailed Description	2931
5.22 base.h File Reference	2931
5.22.1 Detailed Description	2932
5.23 base.h File Reference	2932
5.23.1 Detailed Description	2933
5.24 basic_file.h File Reference	2933
5.24.1 Detailed Description	2933
5.25 basic_ios.h File Reference	2933
5.25.1 Detailed Description	2934
5.26 basic_ios.tcc File Reference	2934
5.26.1 Detailed Description	2934
5.27 basic_iterator.h File Reference	2934
5.27.1 Detailed Description	2934
5.28 basic_string.h File Reference	2934
5.28.1 Detailed Description	2937
5.29 basic_string.tcc File Reference	2937
5.29.1 Detailed Description	2937
5.30 bin_search_tree_.hpp File Reference	2938
5.30.1 Detailed Description	2938

5.31	binary_heap_.hpp File Reference	2938
5.31.1	Detailed Description	2938
5.32	binders.h File Reference	2939
5.32.1	Detailed Description	2939
5.33	binomial_heap_.hpp File Reference	2939
5.33.1	Detailed Description	2939
5.34	binomial_heap_base_.hpp File Reference	2940
5.34.1	Detailed Description	2940
5.35	bitmap_allocator.h File Reference	2940
5.35.1	Detailed Description	2941
5.35.2	Macro Definition Documentation	2941
5.36	bitset File Reference	2941
5.36.1	Detailed Description	2944
5.37	bitset File Reference	2944
5.37.1	Detailed Description	2944
5.38	bitset File Reference	2944
5.38.1	Detailed Description	2945
5.39	bool_set File Reference	2945
5.39.1	Detailed Description	2946
5.40	bool_set.tcc File Reference	2946
5.40.1	Detailed Description	2946
5.41	boost_concept_check.h File Reference	2946
5.41.1	Detailed Description	2947
5.42	branch_policy.hpp File Reference	2947
5.42.1	Detailed Description	2947
5.43	c++0x_warning.h File Reference	2948
5.43.1	Detailed Description	2948
5.44	c++allocator.h File Reference	2948
5.44.1	Detailed Description	2948
5.45	c++config.h File Reference	2948
5.45.1	Detailed Description	2953
5.46	c++io.h File Reference	2953
5.46.1	Detailed Description	2953
5.47	c++locale.h File Reference	2953
5.47.1	Detailed Description	2953
5.48	c++locale_internal.h File Reference	2954
5.48.1	Detailed Description	2954

5.49	cassert File Reference	2954
5.49.1	Detailed Description	2954
5.50	cast.h File Reference	2954
5.50.1	Detailed Description	2955
5.51	cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference	2955
5.51.1	Detailed Description	2955
5.52	cc_ht_map.hpp File Reference	2955
5.52.1	Detailed Description	2955
5.53	ccomplex File Reference	2955
5.53.1	Detailed Description	2956
5.54	ccomplex File Reference	2956
5.54.1	Detailed Description	2956
5.55	cctype File Reference	2956
5.55.1	Detailed Description	2956
5.56	cctype File Reference	2956
5.56.1	Detailed Description	2956
5.57	cerrno File Reference	2957
5.57.1	Detailed Description	2957
5.58	cfenv File Reference	2957
5.58.1	Detailed Description	2957
5.59	cfenv File Reference	2957
5.59.1	Detailed Description	2957
5.60	cfloat File Reference	2957
5.60.1	Detailed Description	2958
5.61	cfloat File Reference	2958
5.61.1	Detailed Description	2958
5.62	char_traits.h File Reference	2958
5.62.1	Detailed Description	2958
5.63	checkers.h File Reference	2959
5.63.1	Detailed Description	2959
5.64	chrono File Reference	2959
5.64.1	Detailed Description	2962
5.65	cinttypes File Reference	2962
5.65.1	Detailed Description	2962
5.66	cinttypes File Reference	2962
5.66.1	Detailed Description	2962
5.67	ciso646 File Reference	2962

5.67.1 Detailed Description	2962
5.68 climits File Reference	2963
5.68.1 Detailed Description	2963
5.69 climits File Reference	2963
5.69.1 Detailed Description	2963
5.70 locale File Reference	2963
5.70.1 Detailed Description	2963
5.71 cmath File Reference	2964
5.71.1 Detailed Description	2967
5.72 cmath File Reference	2967
5.72.1 Detailed Description	2970
5.73 cmp_fn_imps.hpp File Reference	2970
5.73.1 Detailed Description	2970
5.74 codecvt.h File Reference	2970
5.74.1 Detailed Description	2970
5.75 codecvt_specializations.h File Reference	2970
5.75.1 Detailed Description	2971
5.76 compatibility.h File Reference	2971
5.76.1 Detailed Description	2971
5.77 compatibility.h File Reference	2971
5.77.1 Detailed Description	2972
5.78 compiletime_settings.h File Reference	2972
5.78.1 Detailed Description	2972
5.78.2 Macro Definition Documentation	2972
5.79 complex File Reference	2973
5.79.1 Detailed Description	2977
5.80 complex File Reference	2977
5.80.1 Detailed Description	2978
5.81 complex.h File Reference	2978
5.81.1 Detailed Description	2978
5.82 concept_check.h File Reference	2978
5.82.1 Detailed Description	2979
5.83 concurrence.h File Reference	2979
5.83.1 Detailed Description	2979
5.84 cond_dealtor.hpp File Reference	2979
5.84.1 Detailed Description	2980
5.85 cond_key_dtor_entry_dealtor.hpp File Reference	2980

5.85.1 Detailed Description	2980
5.86 condition_variable File Reference	2980
5.86.1 Detailed Description	2981
5.87 const_iterator.hpp File Reference	2981
5.87.1 Detailed Description	2981
5.88 const_iterator.hpp File Reference	2981
5.88.1 Detailed Description	2981
5.89 const_iterator.hpp File Reference	2982
5.89.1 Detailed Description	2982
5.90 constructor_destructor_fn_imps.hpp File Reference	2982
5.90.1 Detailed Description	2982
5.91 constructor_destructor_fn_imps.hpp File Reference	2982
5.91.1 Detailed Description	2982
5.92 constructor_destructor_fn_imps.hpp File Reference	2982
5.93 constructor_destructor_no_store_hash_fn_imps.hpp File Reference	2982
5.93.1 Detailed Description	2982
5.94 constructor_destructor_no_store_hash_fn_imps.hpp File Reference	2982
5.94.1 Detailed Description	2982
5.95 constructor_destructor_store_hash_fn_imps.hpp File Reference	2982
5.95.1 Detailed Description	2983
5.96 constructor_destructor_store_hash_fn_imps.hpp File Reference	2983
5.96.1 Detailed Description	2983
5.97 constructors_destructor_fn_imps.hpp File Reference	2983
5.97.1 Detailed Description	2983
5.98 constructors_destructor_fn_imps.hpp File Reference	2983
5.98.1 Detailed Description	2983
5.99 constructors_destructor_fn_imps.hpp File Reference	2983
5.99.1 Detailed Description	2983
5.100constructors_destructor_fn_imps.hpp File Reference	2983
5.100.1 Detailed Description	2983
5.101constructors_destructor_fn_imps.hpp File Reference	2983
5.101.1 Detailed Description	2983
5.102constructors_destructor_fn_imps.hpp File Reference	2984
5.102.1 Detailed Description	2984
5.103constructors_destructor_fn_imps.hpp File Reference	2984
5.103.1 Detailed Description	2984
5.104constructors_destructor_fn_imps.hpp File Reference	2984

5.104.1 Detailed Description	2984
5.105constructors_destructor_fn_imps.hpp File Reference	2984
5.105.1 Detailed Description	2984
5.106constructors_destructor_fn_imps.hpp File Reference	2984
5.106.1 Detailed Description	2984
5.107constructors_destructor_fn_imps.hpp File Reference	2984
5.107.1 Detailed Description	2984
5.108constructors_destructor_fn_imps.hpp File Reference	2985
5.108.1 Detailed Description	2985
5.109container_base_dispatch.hpp File Reference	2985
5.109.1 Detailed Description	2986
5.110cpp_type_traits.h File Reference	2986
5.110.1 Detailed Description	2986
5.111cpu_defines.h File Reference	2986
5.111.1 Detailed Description	2986
5.112csetjmp File Reference	2986
5.112.1 Detailed Description	2987
5.113csignal File Reference	2987
5.113.1 Detailed Description	2987
5.114cstdarg File Reference	2987
5.114.1 Detailed Description	2987
5.115cstdarg File Reference	2988
5.115.1 Detailed Description	2988
5.116cstdbool File Reference	2988
5.116.1 Detailed Description	2988
5.117cstdbool File Reference	2988
5.117.1 Detailed Description	2988
5.118cstddef File Reference	2988
5.118.1 Detailed Description	2988
5.119cstdint File Reference	2989
5.119.1 Detailed Description	2989
5.120cstdint File Reference	2989
5.120.1 Detailed Description	2989
5.121cstdio File Reference	2989
5.121.1 Detailed Description	2990
5.122cstdio File Reference	2990
5.122.1 Detailed Description	2990

5.123cstdlib File Reference	2990
5.123.1 Detailed Description	2990
5.124cstdlib File Reference	2991
5.124.1 Detailed Description	2991
5.125cstring File Reference	2991
5.125.1 Detailed Description	2991
5.126cmath File Reference	2991
5.126.1 Detailed Description	2991
5.127cmath File Reference	2992
5.127.1 Detailed Description	2992
5.128ctime File Reference	2992
5.128.1 Detailed Description	2992
5.129ctime File Reference	2992
5.129.1 Detailed Description	2992
5.130ctype_base.h File Reference	2992
5.130.1 Detailed Description	2993
5.131ctype_inline.h File Reference	2993
5.131.1 Detailed Description	2993
5.132wchar File Reference	2993
5.132.1 Detailed Description	2993
5.133wchar File Reference	2994
5.133.1 Detailed Description	2994
5.134wctype File Reference	2994
5.134.1 Detailed Description	2994
5.135wctype File Reference	2994
5.135.1 Detailed Description	2995
5.136cxxabi.h File Reference	2995
5.136.1 Detailed Description	2996
5.137cxxabi_forced.h File Reference	2996
5.137.1 Detailed Description	2996
5.138cxxabi_tweaks.h File Reference	2996
5.138.1 Detailed Description	2997
5.139debug.h File Reference	2997
5.139.1 Detailed Description	2998
5.140debug_allocator.h File Reference	2998
5.140.1 Detailed Description	2998
5.141debug_fn_imps.hpp File Reference	2998

5.141.1 Detailed Description	2998
5.142debug_fn_imps.hpp File Reference	2998
5.142.1 Detailed Description	2998
5.143debug_fn_imps.hpp File Reference	2998
5.143.1 Detailed Description	2998
5.144debug_fn_imps.hpp File Reference	2999
5.144.1 Detailed Description	2999
5.145debug_fn_imps.hpp File Reference	2999
5.145.1 Detailed Description	2999
5.146debug_fn_imps.hpp File Reference	2999
5.146.1 Detailed Description	2999
5.147debug_fn_imps.hpp File Reference	2999
5.147.1 Detailed Description	2999
5.148debug_fn_imps.hpp File Reference	2999
5.148.1 Detailed Description	2999
5.149debug_fn_imps.hpp File Reference	2999
5.149.1 Detailed Description	2999
5.150debug_fn_imps.hpp File Reference	2999
5.150.1 Detailed Description	3000
5.151debug_fn_imps.hpp File Reference	3000
5.151.1 Detailed Description	3000
5.152debug_fn_imps.hpp File Reference	3000
5.152.1 Detailed Description	3000
5.153debug_fn_imps.hpp File Reference	3000
5.153.1 Detailed Description	3000
5.154debug_fn_imps.hpp File Reference	3000
5.154.1 Detailed Description	3000
5.155debug_fn_imps.hpp File Reference	3000
5.155.1 Detailed Description	3000
5.156debug_map_base.hpp File Reference	3000
5.156.1 Detailed Description	3000
5.157debug_no_store_hash_fn_imps.hpp File Reference	3001
5.157.1 Detailed Description	3001
5.158debug_no_store_hash_fn_imps.hpp File Reference	3001
5.158.1 Detailed Description	3001
5.159debug_store_hash_fn_imps.hpp File Reference	3001
5.159.1 Detailed Description	3001

5.160debug_store_hash_fn_imps.hpp File Reference	3001
5.160.1 Detailed Description	3001
5.161decimal File Reference	3001
5.161.1 Detailed Description	3011
5.162deque File Reference	3011
5.162.1 Detailed Description	3011
5.163deque File Reference	3011
5.163.1 Detailed Description	3012
5.164deque File Reference	3012
5.164.1 Detailed Description	3012
5.165deque.tcc File Reference	3012
5.165.1 Detailed Description	3013
5.166direct_mask_range_hashing_imp.hpp File Reference	3013
5.166.1 Detailed Description	3013
5.167direct_mod_range_hashing_imp.hpp File Reference	3013
5.167.1 Detailed Description	3013
5.168dynamic_bitset File Reference	3014
5.168.1 Detailed Description	3015
5.169enc_filebuf.h File Reference	3015
5.169.1 Detailed Description	3015
5.170entry_cmp.hpp File Reference	3015
5.170.1 Detailed Description	3016
5.171entry_list_fn_imps.hpp File Reference	3016
5.171.1 Detailed Description	3016
5.172entry_metadata_base.hpp File Reference	3016
5.172.1 Detailed Description	3016
5.173entry_pred.hpp File Reference	3016
5.173.1 Detailed Description	3016
5.174eq_by_less.hpp File Reference	3017
5.174.1 Detailed Description	3017
5.175equally_split.h File Reference	3017
5.175.1 Detailed Description	3017
5.176erase_fn_imps.hpp File Reference	3017
5.176.1 Detailed Description	3017
5.177erase_fn_imps.hpp File Reference	3018
5.177.1 Detailed Description	3018
5.178erase_fn_imps.hpp File Reference	3018

5.178.1 Detailed Description	3018
5.179erase_fn_imps.hpp File Reference	3018
5.179.1 Detailed Description	3018
5.180erase_fn_imps.hpp File Reference	3018
5.180.1 Detailed Description	3018
5.181erase_fn_imps.hpp File Reference	3018
5.181.1 Detailed Description	3018
5.182erase_fn_imps.hpp File Reference	3018
5.182.1 Detailed Description	3018
5.183erase_fn_imps.hpp File Reference	3018
5.183.1 Detailed Description	3019
5.184erase_fn_imps.hpp File Reference	3019
5.184.1 Detailed Description	3019
5.185erase_fn_imps.hpp File Reference	3019
5.185.1 Detailed Description	3019
5.186erase_fn_imps.hpp File Reference	3019
5.186.1 Detailed Description	3019
5.187erase_fn_imps.hpp File Reference	3019
5.187.1 Detailed Description	3019
5.188erase_fn_imps.hpp File Reference	3019
5.188.1 Detailed Description	3019
5.189erase_fn_imps.hpp File Reference	3019
5.189.1 Detailed Description	3019
5.190erase_no_store_hash_fn_imps.hpp File Reference	3020
5.190.1 Detailed Description	3020
5.191erase_no_store_hash_fn_imps.hpp File Reference	3020
5.191.1 Detailed Description	3020
5.192erase_store_hash_fn_imps.hpp File Reference	3020
5.192.1 Detailed Description	3020
5.193erase_store_hash_fn_imps.hpp File Reference	3020
5.193.1 Detailed Description	3020
5.194error_constants.h File Reference	3020
5.194.1 Detailed Description	3020
5.195exception File Reference	3021
5.195.1 Detailed Description	3021
5.196exception.hpp File Reference	3021
5.196.1 Detailed Description	3022

5.197exception_defines.h File Reference	3022
5.197.1 Detailed Description	3022
5.198exception_ptr.h File Reference	3022
5.198.1 Detailed Description	3023
5.199extc++.h File Reference	3023
5.199.1 Detailed Description	3023
5.200extptr_allocator.h File Reference	3023
5.200.1 Detailed Description	3023
5.201features.h File Reference	3024
5.201.1 Detailed Description	3024
5.201.2 Macro Definition Documentation	3024
5.202fenv.h File Reference	3026
5.202.1 Detailed Description	3026
5.203find.h File Reference	3026
5.203.1 Detailed Description	3026
5.204find_fn_imps.hpp File Reference	3026
5.204.1 Detailed Description	3026
5.205find_fn_imps.hpp File Reference	3027
5.205.1 Detailed Description	3027
5.206find_fn_imps.hpp File Reference	3027
5.206.1 Detailed Description	3027
5.207find_fn_imps.hpp File Reference	3027
5.207.1 Detailed Description	3027
5.208find_fn_imps.hpp File Reference	3027
5.208.1 Detailed Description	3027
5.209find_fn_imps.hpp File Reference	3027
5.209.1 Detailed Description	3027
5.210find_fn_imps.hpp File Reference	3027
5.210.1 Detailed Description	3027
5.211find_fn_imps.hpp File Reference	3027
5.211.1 Detailed Description	3028
5.212find_fn_imps.hpp File Reference	3028
5.212.1 Detailed Description	3028
5.213find_fn_imps.hpp File Reference	3028
5.213.1 Detailed Description	3028
5.214find_fn_imps.hpp File Reference	3028
5.214.1 Detailed Description	3028

5.215	find_no_store_hash_fn_imps.hpp File Reference	3028
5.215.1	Detailed Description	3028
5.216	find_selectors.h File Reference	3028
5.216.1	Detailed Description	3029
5.217	find_store_hash_fn_imps.hpp File Reference	3029
5.217.1	Detailed Description	3029
5.218	find_store_hash_fn_imps.hpp File Reference	3029
5.218.1	Detailed Description	3029
5.219	for_each.h File Reference	3029
5.219.1	Detailed Description	3029
5.220	for_each_selectors.h File Reference	3030
5.220.1	Detailed Description	3031
5.221	formatter.h File Reference	3031
5.221.1	Detailed Description	3031
5.222	forward_list File Reference	3031
5.222.1	Detailed Description	3032
5.223	forward_list File Reference	3032
5.223.1	Detailed Description	3032
5.224	forward_list File Reference	3033
5.224.1	Detailed Description	3033
5.225	forward_list.h File Reference	3033
5.225.1	Detailed Description	3034
5.226	forward_list.tcc File Reference	3034
5.226.1	Detailed Description	3035
5.227	fstream File Reference	3035
5.227.1	Detailed Description	3035
5.228	fstream.tcc File Reference	3035
5.228.1	Detailed Description	3036
5.229	functexcept.h File Reference	3036
5.229.1	Detailed Description	3036
5.230	functional File Reference	3037
5.230.1	Detailed Description	3041
5.231	functional File Reference	3041
5.231.1	Detailed Description	3042
5.232	functional_hash.h File Reference	3042
5.232.1	Detailed Description	3043
5.233	functions.h File Reference	3044

5.233.1 Detailed Description	3046
5.234future File Reference	3046
5.234.1 Detailed Description	3047
5.235gp_ht_map_.hpp File Reference	3047
5.235.1 Detailed Description	3048
5.236gslice.h File Reference	3048
5.236.1 Detailed Description	3048
5.237gslice_array.h File Reference	3048
5.237.1 Detailed Description	3049
5.238hash_bytes.h File Reference	3049
5.238.1 Detailed Description	3049
5.239hash_eq_fn.hpp File Reference	3049
5.239.1 Detailed Description	3049
5.240hash_exponential_size_policy_imp.hpp File Reference	3050
5.240.1 Detailed Description	3050
5.241hash_fun.h File Reference	3050
5.241.1 Detailed Description	3050
5.242hash_load_check_resize_trigger_imp.hpp File Reference	3050
5.242.1 Detailed Description	3050
5.243hash_load_check_resize_trigger_size_base.hpp File Reference	3050
5.243.1 Detailed Description	3051
5.244hash_map File Reference	3051
5.244.1 Detailed Description	3051
5.245hash_policy.hpp File Reference	3052
5.245.1 Detailed Description	3053
5.246hash_prime_size_policy_imp.hpp File Reference	3053
5.246.1 Detailed Description	3053
5.247hash_set File Reference	3053
5.247.1 Detailed Description	3054
5.248hash_standard_resize_policy_imp.hpp File Reference	3054
5.248.1 Detailed Description	3054
5.249hashtable.h File Reference	3054
5.249.1 Detailed Description	3055
5.250hashtable.h File Reference	3055
5.250.1 Detailed Description	3055
5.251hashtable_policy.h File Reference	3055
5.251.1 Detailed Description	3058

5.252indirect_array.h File Reference	3058
5.252.1 Detailed Description	3058
5.253info_fn_imps.hpp File Reference	3058
5.253.1 Detailed Description	3058
5.254info_fn_imps.hpp File Reference	3059
5.254.1 Detailed Description	3059
5.255info_fn_imps.hpp File Reference	3059
5.255.1 Detailed Description	3059
5.256info_fn_imps.hpp File Reference	3059
5.256.1 Detailed Description	3059
5.257info_fn_imps.hpp File Reference	3059
5.257.1 Detailed Description	3059
5.258info_fn_imps.hpp File Reference	3059
5.258.1 Detailed Description	3059
5.259info_fn_imps.hpp File Reference	3059
5.259.1 Detailed Description	3059
5.260info_fn_imps.hpp File Reference	3059
5.260.1 Detailed Description	3060
5.261info_fn_imps.hpp File Reference	3060
5.261.1 Detailed Description	3060
5.262info_fn_imps.hpp File Reference	3060
5.262.1 Detailed Description	3060
5.263initializer_list File Reference	3060
5.263.1 Detailed Description	3060
5.264insert_fn_imps.hpp File Reference	3060
5.264.1 Detailed Description	3061
5.265insert_fn_imps.hpp File Reference	3061
5.265.1 Detailed Description	3061
5.266insert_fn_imps.hpp File Reference	3061
5.266.1 Detailed Description	3061
5.267insert_fn_imps.hpp File Reference	3061
5.267.1 Detailed Description	3061
5.268insert_fn_imps.hpp File Reference	3061
5.268.1 Detailed Description	3061
5.269insert_fn_imps.hpp File Reference	3061
5.269.1 Detailed Description	3061
5.270insert_fn_imps.hpp File Reference	3061

5.270.1 Detailed Description	3061
5.271insert_fn_imps.hpp File Reference	3062
5.271.1 Detailed Description	3062
5.272insert_fn_imps.hpp File Reference	3062
5.272.1 Detailed Description	3062
5.273insert_fn_imps.hpp File Reference	3062
5.273.1 Detailed Description	3062
5.274insert_fn_imps.hpp File Reference	3062
5.274.1 Detailed Description	3062
5.275insert_fn_imps.hpp File Reference	3062
5.275.1 Detailed Description	3062
5.276insert_fn_imps.hpp File Reference	3062
5.276.1 Detailed Description	3062
5.277insert_join_fn_imps.hpp File Reference	3063
5.277.1 Detailed Description	3063
5.278insert_no_store_hash_fn_imps.hpp File Reference	3063
5.278.1 Detailed Description	3063
5.279insert_no_store_hash_fn_imps.hpp File Reference	3063
5.279.1 Detailed Description	3063
5.280insert_store_hash_fn_imps.hpp File Reference	3063
5.280.1 Detailed Description	3063
5.281insert_store_hash_fn_imps.hpp File Reference	3063
5.281.1 Detailed Description	3063
5.282iomanip File Reference	3063
5.282.1 Detailed Description	3065
5.283ios File Reference	3065
5.283.1 Detailed Description	3065
5.284ios_base.h File Reference	3065
5.284.1 Detailed Description	3066
5.285iosfwd File Reference	3067
5.285.1 Detailed Description	3068
5.286iostream File Reference	3068
5.286.1 Detailed Description	3068
5.287istream File Reference	3068
5.287.1 Detailed Description	3069
5.288istream.tcc File Reference	3069
5.288.1 Detailed Description	3070

5.289	iterator File Reference	3070
5.289.1	Detailed Description	3070
5.290	iterator File Reference	3070
5.290.1	Detailed Description	3071
5.291	iterator.h File Reference	3071
5.291.1	Detailed Description	3071
5.292	iterator.hpp File Reference	3071
5.292.1	Detailed Description	3071
5.293	iterator_fn_imps.hpp File Reference	3071
5.293.1	Detailed Description	3071
5.294	iterator_tracker.h File Reference	3072
5.294.1	Detailed Description	3073
5.295	iterators_fn_imps.hpp File Reference	3073
5.295.1	Detailed Description	3073
5.296	iterators_fn_imps.hpp File Reference	3073
5.296.1	Detailed Description	3073
5.297	iterators_fn_imps.hpp File Reference	3073
5.297.1	Detailed Description	3073
5.298	iterators_fn_imps.hpp File Reference	3073
5.298.1	Detailed Description	3073
5.299	iterators_fn_imps.hpp File Reference	3073
5.299.1	Detailed Description	3073
5.300	iterators_fn_imps.hpp File Reference	3074
5.300.1	Detailed Description	3074
5.301	iterators_fn_imps.hpp File Reference	3074
5.301.1	Detailed Description	3074
5.302	left_child_next_sibling_heap_.hpp File Reference	3074
5.302.1	Detailed Description	3074
5.303	limits File Reference	3074
5.303.1	Detailed Description	3076
5.304	linear_probe_fn_imp.hpp File Reference	3076
5.304.1	Detailed Description	3076
5.305	list File Reference	3076
5.305.1	Detailed Description	3076
5.306	list File Reference	3076
5.306.1	Detailed Description	3077
5.307	list File Reference	3077

5.307.1 Detailed Description	3078
5.308list.tcc File Reference	3078
5.308.1 Detailed Description	3078
5.309list_partition.h File Reference	3078
5.309.1 Detailed Description	3079
5.310list_update_policy.hpp File Reference	3079
5.310.1 Detailed Description	3079
5.311locale File Reference	3079
5.311.1 Detailed Description	3079
5.312locale_classes.h File Reference	3079
5.312.1 Detailed Description	3080
5.313locale_classes.tcc File Reference	3080
5.313.1 Detailed Description	3080
5.314locale_facets.h File Reference	3081
5.314.1 Detailed Description	3082
5.315locale_facets.tcc File Reference	3082
5.315.1 Detailed Description	3083
5.316locale_facets_nonio.h File Reference	3083
5.316.1 Detailed Description	3084
5.317locale_facets_nonio.tcc File Reference	3084
5.317.1 Detailed Description	3084
5.318localefwd.h File Reference	3084
5.318.1 Detailed Description	3086
5.319losertree.h File Reference	3087
5.319.1 Detailed Description	3087
5.320lu_counter_metadata.hpp File Reference	3088
5.320.1 Detailed Description	3088
5.321lu_map_.hpp File Reference	3088
5.321.1 Detailed Description	3088
5.322macros.h File Reference	3089
5.322.1 Detailed Description	3089
5.322.2 Macro Definition Documentation	3089
5.323malloc_allocator.h File Reference	3091
5.323.1 Detailed Description	3092
5.324map File Reference	3092
5.324.1 Detailed Description	3092
5.325map File Reference	3092

5.325.1 Detailed Description	3092
5.326map File Reference	3092
5.326.1 Detailed Description	3092
5.327map.h File Reference	3093
5.327.1 Detailed Description	3093
5.328map.h File Reference	3093
5.328.1 Detailed Description	3094
5.329mask_array.h File Reference	3094
5.329.1 Detailed Description	3095
5.330mask_based_range_hashing.hpp File Reference	3095
5.330.1 Detailed Description	3095
5.331memory File Reference	3095
5.331.1 Detailed Description	3095
5.332memory File Reference	3095
5.332.1 Detailed Description	3096
5.333merge.h File Reference	3096
5.333.1 Detailed Description	3097
5.334messages_members.h File Reference	3097
5.334.1 Detailed Description	3097
5.335mod_based_range_hashing.hpp File Reference	3097
5.335.1 Detailed Description	3097
5.336move.h File Reference	3097
5.336.1 Detailed Description	3098
5.337mt_allocator.h File Reference	3098
5.337.1 Detailed Description	3099
5.338multimap.h File Reference	3099
5.338.1 Detailed Description	3100
5.339multimap.h File Reference	3100
5.339.1 Detailed Description	3101
5.340multiseg_selection.h File Reference	3101
5.340.1 Detailed Description	3102
5.341multiset.h File Reference	3102
5.341.1 Detailed Description	3102
5.342multiset.h File Reference	3103
5.342.1 Detailed Description	3103
5.343multiway_merge.h File Reference	3103
5.343.1 Detailed Description	3106

5.343.2 Macro Definition Documentation	3106
5.344multiway_mergesort.h File Reference	3107
5.344.1 Detailed Description	3107
5.345mutex File Reference	3107
5.345.1 Detailed Description	3109
5.346nested_exception.h File Reference	3109
5.346.1 Detailed Description	3109
5.347new File Reference	3109
5.347.1 Detailed Description	3110
5.347.2 Function Documentation	3110
5.348new_allocator.h File Reference	3113
5.348.1 Detailed Description	3114
5.349node.hpp File Reference	3114
5.349.1 Detailed Description	3114
5.350node.hpp File Reference	3114
5.350.1 Detailed Description	3114
5.351node.hpp File Reference	3115
5.351.1 Detailed Description	3115
5.352node_iterators.hpp File Reference	3115
5.352.1 Detailed Description	3115
5.353node_iterators.hpp File Reference	3115
5.353.1 Detailed Description	3116
5.354node_metadata_selector.hpp File Reference	3116
5.354.1 Detailed Description	3116
5.355node_metadata_selector.hpp File Reference	3116
5.355.1 Detailed Description	3117
5.356null_node_metadata.hpp File Reference	3117
5.356.1 Detailed Description	3117
5.357numeric File Reference	3117
5.357.1 Detailed Description	3117
5.358numeric File Reference	3117
5.358.1 Detailed Description	3118
5.359numeric File Reference	3118
5.359.1 Detailed Description	3120
5.360numeric_traits.h File Reference	3120
5.360.1 Detailed Description	3120
5.361numeric_fwd.h File Reference	3120

5.361.1 Detailed Description	3122
5.362omp_loop.h File Reference	3122
5.362.1 Detailed Description	3122
5.363omp_loop_static.h File Reference	3123
5.363.1 Detailed Description	3123
5.364opt_random.h File Reference	3123
5.364.1 Detailed Description	3123
5.365order_statistics_imp.hpp File Reference	3123
5.365.1 Detailed Description	3123
5.366order_statistics_imp.hpp File Reference	3123
5.366.1 Detailed Description	3123
5.367os_defines.h File Reference	3124
5.367.1 Detailed Description	3124
5.368ostream File Reference	3124
5.368.1 Detailed Description	3125
5.369ostream.tcc File Reference	3125
5.369.1 Detailed Description	3125
5.370ostream_insert.h File Reference	3126
5.370.1 Detailed Description	3126
5.371ov_tree_map_.hpp File Reference	3126
5.371.1 Detailed Description	3126
5.372pairing_heap_.hpp File Reference	3127
5.372.1 Detailed Description	3127
5.373par_loop.h File Reference	3127
5.373.1 Detailed Description	3127
5.374parallel.h File Reference	3127
5.374.1 Detailed Description	3127
5.375partial_sum.h File Reference	3128
5.375.1 Detailed Description	3128
5.376partition.h File Reference	3128
5.376.1 Detailed Description	3129
5.376.2 Macro Definition Documentation	3129
5.377pat_trie_.hpp File Reference	3129
5.377.1 Detailed Description	3129
5.378pat_trie_base.hpp File Reference	3129
5.378.1 Detailed Description	3130
5.379pod_char_traits.h File Reference	3131

5.379.1 Detailed Description	3131
5.380point_const_iterator.hpp File Reference	3131
5.380.1 Detailed Description	3131
5.381point_const_iterator.hpp File Reference	3131
5.381.1 Detailed Description	3132
5.382point_const_iterator.hpp File Reference	3132
5.382.1 Detailed Description	3132
5.383point_iterator.hpp File Reference	3132
5.383.1 Detailed Description	3132
5.384point_iterators.hpp File Reference	3132
5.384.1 Detailed Description	3133
5.385pointer.h File Reference	3133
5.385.1 Detailed Description	3135
5.386policy_access_fn_imps.hpp File Reference	3135
5.386.1 Detailed Description	3135
5.387policy_access_fn_imps.hpp File Reference	3135
5.387.1 Detailed Description	3135
5.388policy_access_fn_imps.hpp File Reference	3135
5.388.1 Detailed Description	3135
5.389policy_access_fn_imps.hpp File Reference	3136
5.389.1 Detailed Description	3136
5.390policy_access_fn_imps.hpp File Reference	3136
5.390.1 Detailed Description	3136
5.391policy_access_fn_imps.hpp File Reference	3136
5.391.1 Detailed Description	3136
5.392policy_access_fn_imps.hpp File Reference	3136
5.392.1 Detailed Description	3136
5.393pool_allocator.h File Reference	3136
5.393.1 Detailed Description	3137
5.394postypes.h File Reference	3137
5.394.1 Detailed Description	3137
5.395prefix_search_node_update_imp.hpp File Reference	3137
5.395.1 Detailed Description	3137
5.396priority_queue.hpp File Reference	3138
5.396.1 Detailed Description	3138
5.397priority_queue_base_dispatch.hpp File Reference	3138
5.397.1 Detailed Description	3138

5.398probe_fn_base.hpp File Reference	3139
5.398.1 Detailed Description	3139
5.399profiler.h File Reference	3139
5.399.1 Detailed Description	3141
5.400profiler_algos.h File Reference	3141
5.400.1 Detailed Description	3142
5.401profiler_container_size.h File Reference	3142
5.401.1 Detailed Description	3142
5.402profiler_hash_func.h File Reference	3142
5.402.1 Detailed Description	3143
5.403profiler_hashtable_size.h File Reference	3143
5.403.1 Detailed Description	3143
5.404profiler_list_to_slist.h File Reference	3143
5.404.1 Detailed Description	3144
5.405profiler_list_to_vector.h File Reference	3144
5.405.1 Detailed Description	3144
5.406profiler_map_to_unordered_map.h File Reference	3144
5.406.1 Detailed Description	3145
5.407profiler_node.h File Reference	3145
5.407.1 Detailed Description	3146
5.408profiler_state.h File Reference	3146
5.408.1 Detailed Description	3146
5.409profiler_trace.h File Reference	3147
5.409.1 Detailed Description	3149
5.410profiler_vector_size.h File Reference	3149
5.410.1 Detailed Description	3149
5.411profiler_vector_to_list.h File Reference	3149
5.411.1 Detailed Description	3150
5.412ptr_traits.h File Reference	3150
5.412.1 Detailed Description	3150
5.413quadratic_probe_fn_imp.hpp File Reference	3150
5.413.1 Detailed Description	3150
5.414queue File Reference	3150
5.414.1 Detailed Description	3151
5.415queue.h File Reference	3151
5.415.1 Detailed Description	3151
5.415.2 Macro Definition Documentation	3151

5.416	quicksort.h File Reference	3151
5.416.1	Detailed Description	3152
5.417	r_erase_fn_imps.hpp File Reference	3152
5.417.1	Detailed Description	3152
5.418	r_erase_fn_imps.hpp File Reference	3152
5.418.1	Detailed Description	3152
5.419	random File Reference	3152
5.419.1	Detailed Description	3152
5.420	random.h File Reference	3152
5.420.1	Detailed Description	3158
5.421	random.tcc File Reference	3158
5.421.1	Detailed Description	3162
5.422	random.tcc File Reference	3162
5.422.1	Detailed Description	3164
5.423	random_number.h File Reference	3164
5.423.1	Detailed Description	3164
5.424	random_shuffle.h File Reference	3165
5.424.1	Detailed Description	3165
5.425	range_access.h File Reference	3165
5.425.1	Detailed Description	3166
5.426	ranged_hash_fn.hpp File Reference	3166
5.426.1	Detailed Description	3167
5.427	ranged_probe_fn.hpp File Reference	3167
5.427.1	Detailed Description	3167
5.428	ratio File Reference	3167
5.428.1	Detailed Description	3168
5.429	ratio File Reference	3168
5.429.1	Detailed Description	3168
5.430	rb_tree File Reference	3168
5.430.1	Detailed Description	3168
5.431	rb_tree_.hpp File Reference	3169
5.431.1	Detailed Description	3169
5.432	rc.hpp File Reference	3169
5.432.1	Detailed Description	3169
5.433	rc_binomial_heap_.hpp File Reference	3169
5.433.1	Detailed Description	3170
5.434	rc_string_base.h File Reference	3170

5.434.1 Detailed Description	3170
5.435regex File Reference	3170
5.435.1 Detailed Description	3170
5.436regex.h File Reference	3170
5.436.1 Detailed Description	3175
5.437regex_compiler.h File Reference	3175
5.437.1 Detailed Description	3176
5.438regex_constants.h File Reference	3176
5.438.1 Detailed Description	3177
5.439regex_cursor.h File Reference	3177
5.439.1 Detailed Description	3178
5.440regex_error.h File Reference	3178
5.440.1 Detailed Description	3178
5.441regex_grep_matcher.h File Reference	3179
5.441.1 Detailed Description	3179
5.442regex_grep_matcher.tcc File Reference	3179
5.442.1 Detailed Description	3179
5.443regex_nfa.h File Reference	3179
5.443.1 Detailed Description	3181
5.444regex_nfa.tcc File Reference	3181
5.444.1 Detailed Description	3181
5.445resize_fn_imps.hpp File Reference	3181
5.445.1 Detailed Description	3181
5.446resize_fn_imps.hpp File Reference	3181
5.446.1 Detailed Description	3181
5.447resize_no_store_hash_fn_imps.hpp File Reference	3181
5.447.1 Detailed Description	3181
5.448resize_no_store_hash_fn_imps.hpp File Reference	3181
5.448.1 Detailed Description	3181
5.449resize_policy.hpp File Reference	3182
5.449.1 Detailed Description	3182
5.450resize_store_hash_fn_imps.hpp File Reference	3182
5.450.1 Detailed Description	3182
5.451resize_store_hash_fn_imps.hpp File Reference	3182
5.451.1 Detailed Description	3182
5.452rope File Reference	3182
5.452.1 Detailed Description	3185

5.453ropeimpl.h File Reference	3186
5.453.1 Detailed Description	3186
5.454rotate_fn_imps.hpp File Reference	3186
5.454.1 Detailed Description	3186
5.455rotate_fn_imps.hpp File Reference	3186
5.455.1 Detailed Description	3186
5.456safe_base.h File Reference	3186
5.456.1 Detailed Description	3187
5.457safe_iterator.h File Reference	3187
5.457.1 Detailed Description	3189
5.458safe_iterator.tcc File Reference	3189
5.458.1 Detailed Description	3189
5.459safe_local_iterator.h File Reference	3189
5.459.1 Detailed Description	3190
5.460safe_local_iterator.tcc File Reference	3190
5.460.1 Detailed Description	3190
5.461safe_sequence.h File Reference	3190
5.461.1 Detailed Description	3190
5.462safe_sequence.tcc File Reference	3190
5.462.1 Detailed Description	3191
5.463safe_unordered_base.h File Reference	3191
5.463.1 Detailed Description	3191
5.464safe_unordered_container.h File Reference	3191
5.464.1 Detailed Description	3191
5.465safe_unordered_container.tcc File Reference	3192
5.465.1 Detailed Description	3192
5.466sample_probe_fn.hpp File Reference	3192
5.466.1 Detailed Description	3192
5.467sample_range_hashing.hpp File Reference	3192
5.467.1 Detailed Description	3192
5.468sample_ranged_hash_fn.hpp File Reference	3193
5.468.1 Detailed Description	3193
5.469sample_ranged_probe_fn.hpp File Reference	3193
5.469.1 Detailed Description	3193
5.470sample_resize_policy.hpp File Reference	3193
5.470.1 Detailed Description	3193
5.471sample_resize_trigger.hpp File Reference	3194

5.471.1 Detailed Description	3194
5.472sample_size_policy.hpp File Reference	3194
5.472.1 Detailed Description	3194
5.473sample_tree_node_update.hpp File Reference	3194
5.473.1 Detailed Description	3194
5.474sample_trie_access_traits.hpp File Reference	3195
5.474.1 Detailed Description	3195
5.475sample_trie_node_update.hpp File Reference	3195
5.475.1 Detailed Description	3195
5.476sample_update_policy.hpp File Reference	3195
5.476.1 Detailed Description	3195
5.477scoped_allocator File Reference	3196
5.477.1 Detailed Description	3196
5.478search.h File Reference	3196
5.478.1 Detailed Description	3197
5.479set File Reference	3197
5.479.1 Detailed Description	3197
5.480set File Reference	3197
5.480.1 Detailed Description	3197
5.481set File Reference	3197
5.481.1 Detailed Description	3197
5.482set.h File Reference	3197
5.482.1 Detailed Description	3198
5.483set.h File Reference	3198
5.483.1 Detailed Description	3199
5.484set_operations.h File Reference	3199
5.484.1 Detailed Description	3200
5.485settings.h File Reference	3200
5.485.1 Detailed Description	3200
5.485.2 parallelization_decision	3200
5.485.3 Macro Definition Documentation	3201
5.486shared_ptr.h File Reference	3201
5.486.1 Detailed Description	3203
5.487shared_ptr_base.h File Reference	3203
5.487.1 Detailed Description	3205
5.488size_fn_imps.hpp File Reference	3205
5.488.1 Detailed Description	3205

5.489	slice_array.h File Reference	3205
5.489.1	Detailed Description	3205
5.490	slist File Reference	3205
5.490.1	Detailed Description	3206
5.491	sort.h File Reference	3206
5.491.1	Detailed Description	3207
5.492	splay_fn_imps.hpp File Reference	3207
5.492.1	Detailed Description	3207
5.493	splay_tree_.hpp File Reference	3207
5.493.1	Detailed Description	3208
5.494	split_fn_imps.hpp File Reference	3208
5.494.1	Detailed Description	3208
5.495	split_join_fn_imps.hpp File Reference	3208
5.495.1	Detailed Description	3208
5.496	split_join_fn_imps.hpp File Reference	3208
5.496.1	Detailed Description	3208
5.497	split_join_fn_imps.hpp File Reference	3208
5.497.1	Detailed Description	3208
5.498	split_join_fn_imps.hpp File Reference	3208
5.498.1	Detailed Description	3208
5.499	split_join_fn_imps.hpp File Reference	3209
5.499.1	Detailed Description	3209
5.500	split_join_fn_imps.hpp File Reference	3209
5.500.1	Detailed Description	3209
5.501	split_join_fn_imps.hpp File Reference	3209
5.501.1	Detailed Description	3209
5.502	split_join_fn_imps.hpp File Reference	3209
5.502.1	Detailed Description	3209
5.503	split_join_fn_imps.hpp File Reference	3209
5.503.1	Detailed Description	3209
5.504	sso_string_base.h File Reference	3209
5.504.1	Detailed Description	3209
5.505	sstream File Reference	3210
5.505.1	Detailed Description	3210
5.506	sstream.tcc File Reference	3210
5.506.1	Detailed Description	3210
5.507	stack File Reference	3210

5.507.1 Detailed Description	3211
5.508standard_policies.hpp File Reference	3211
5.508.1 Detailed Description	3211
5.509stdc++.h File Reference	3212
5.509.1 Detailed Description	3212
5.510stdexcept File Reference	3212
5.510.1 Detailed Description	3212
5.511stdio_filebuf.h File Reference	3212
5.511.1 Detailed Description	3213
5.512stdio_sync_filebuf.h File Reference	3213
5.512.1 Detailed Description	3213
5.513stdtr1c++.h File Reference	3213
5.513.1 Detailed Description	3213
5.514stl_algo.h File Reference	3213
5.514.1 Detailed Description	3222
5.515stl_algobase.h File Reference	3223
5.515.1 Detailed Description	3225
5.516stl_bvector.h File Reference	3225
5.516.1 Detailed Description	3226
5.517stl_construct.h File Reference	3226
5.517.1 Detailed Description	3226
5.518stl_deque.h File Reference	3226
5.518.1 Detailed Description	3229
5.518.2 Macro Definition Documentation	3229
5.519stl_function.h File Reference	3229
5.519.1 Detailed Description	3231
5.520stl_heap.h File Reference	3231
5.520.1 Detailed Description	3232
5.521stl_iterator.h File Reference	3233
5.521.1 Detailed Description	3235
5.522stl_iterator_base_funcs.h File Reference	3236
5.522.1 Detailed Description	3236
5.523stl_iterator_base_types.h File Reference	3237
5.523.1 Detailed Description	3237
5.524stl_list.h File Reference	3238
5.524.1 Detailed Description	3238
5.525stl_map.h File Reference	3239

5.525.1 Detailed Description	3239
5.526stl_multimap.h File Reference	3239
5.526.1 Detailed Description	3240
5.527stl_multiset.h File Reference	3240
5.527.1 Detailed Description	3241
5.528stl_numeric.h File Reference	3241
5.528.1 Detailed Description	3242
5.529stl_pair.h File Reference	3242
5.529.1 Detailed Description	3243
5.530stl_queue.h File Reference	3243
5.530.1 Detailed Description	3243
5.531stl_raw_storage_iter.h File Reference	3244
5.531.1 Detailed Description	3244
5.532stl_relops.h File Reference	3244
5.532.1 Detailed Description	3244
5.533stl_set.h File Reference	3245
5.533.1 Detailed Description	3245
5.534stl_stack.h File Reference	3245
5.534.1 Detailed Description	3246
5.535stl_tempbuf.h File Reference	3246
5.535.1 Detailed Description	3246
5.536stl_tree.h File Reference	3247
5.536.1 Detailed Description	3248
5.537stl_uninitialized.h File Reference	3248
5.537.1 Detailed Description	3249
5.538stl_vector.h File Reference	3249
5.538.1 Detailed Description	3250
5.539stream_iterator.h File Reference	3250
5.539.1 Detailed Description	3250
5.540streambuf File Reference	3251
5.540.1 Detailed Description	3251
5.541streambuf.tcc File Reference	3251
5.541.1 Detailed Description	3252
5.542streambuf_iterator.h File Reference	3252
5.542.1 Detailed Description	3253
5.543string File Reference	3253
5.543.1 Detailed Description	3253

5.544string File Reference	3253
5.544.1 Detailed Description	3255
5.545string_conversions.h File Reference	3256
5.545.1 Detailed Description	3256
5.546stringfwd.h File Reference	3256
5.546.1 Detailed Description	3256
5.547strstream File Reference	3257
5.547.1 Detailed Description	3257
5.548synth_access_traits.hpp File Reference	3257
5.548.1 Detailed Description	3257
5.549system_error File Reference	3257
5.549.1 Detailed Description	3258
5.550tag_and_trait.hpp File Reference	3259
5.550.1 Detailed Description	3261
5.551tags.h File Reference	3261
5.551.1 Detailed Description	3262
5.552tgmth.h File Reference	3262
5.552.1 Detailed Description	3262
5.553thin_heap_.hpp File Reference	3262
5.553.1 Detailed Description	3263
5.554thread File Reference	3263
5.554.1 Detailed Description	3263
5.555throw_allocator.h File Reference	3264
5.555.1 Detailed Description	3265
5.556time_members.h File Reference	3265
5.556.1 Detailed Description	3266
5.557trace_fn_imps.hpp File Reference	3266
5.557.1 Detailed Description	3266
5.558trace_fn_imps.hpp File Reference	3266
5.558.1 Detailed Description	3266
5.559trace_fn_imps.hpp File Reference	3266
5.559.1 Detailed Description	3266
5.560trace_fn_imps.hpp File Reference	3266
5.560.1 Detailed Description	3266
5.561trace_fn_imps.hpp File Reference	3266
5.561.1 Detailed Description	3266
5.562trace_fn_imps.hpp File Reference	3266

5.562.1 Detailed Description	3267
5.563trace_fn_imps.hpp File Reference	3267
5.563.1 Detailed Description	3267
5.564trace_fn_imps.hpp File Reference	3267
5.564.1 Detailed Description	3267
5.565traits.hpp File Reference	3267
5.565.1 Detailed Description	3267
5.566traits.hpp File Reference	3267
5.566.1 Detailed Description	3268
5.567traits.hpp File Reference	3268
5.567.1 Detailed Description	3268
5.568traits.hpp File Reference	3268
5.568.1 Detailed Description	3269
5.569traits.hpp File Reference	3269
5.569.1 Detailed Description	3269
5.570traits.hpp File Reference	3269
5.570.1 Detailed Description	3269
5.571tree_policy.hpp File Reference	3269
5.571.1 Detailed Description	3270
5.572tree_trace_base.hpp File Reference	3270
5.572.1 Detailed Description	3270
5.573trie_policy.hpp File Reference	3270
5.573.1 Detailed Description	3271
5.574trie_policy_base.hpp File Reference	3271
5.574.1 Detailed Description	3271
5.575trie_string_access_traits_imp.hpp File Reference	3271
5.575.1 Detailed Description	3271
5.576tuple File Reference	3271
5.576.1 Detailed Description	3273
5.577type_traits File Reference	3273
5.577.1 Detailed Description	3277
5.577.2 Macro Definition Documentation	3277
5.578type_traits File Reference	3278
5.578.1 Detailed Description	3278
5.579type_traits.h File Reference	3278
5.579.1 Detailed Description	3278
5.580type_utils.hpp File Reference	3279

5.580.1 Detailed Description	3279
5.581typeindex File Reference	3279
5.581.1 Detailed Description	3279
5.582typeinfo File Reference	3280
5.582.1 Detailed Description	3280
5.583typelist.h File Reference	3280
5.583.1 Detailed Description	3281
5.584types.h File Reference	3281
5.584.1 Detailed Description	3282
5.585types_traits.hpp File Reference	3282
5.585.1 Detailed Description	3283
5.586unique_copy.h File Reference	3283
5.586.1 Detailed Description	3283
5.587unique_ptr.h File Reference	3283
5.587.1 Detailed Description	3284
5.588unordered_map File Reference	3284
5.588.1 Detailed Description	3285
5.589unordered_map File Reference	3285
5.589.1 Detailed Description	3285
5.590unordered_map File Reference	3286
5.590.1 Detailed Description	3286
5.591unordered_map.h File Reference	3287
5.591.1 Detailed Description	3288
5.592unordered_set File Reference	3288
5.592.1 Detailed Description	3288
5.593unordered_set File Reference	3288
5.593.1 Detailed Description	3289
5.594unordered_set File Reference	3289
5.594.1 Detailed Description	3290
5.595unordered_set.h File Reference	3290
5.595.1 Detailed Description	3291
5.596update_fn_imps.hpp File Reference	3291
5.596.1 Detailed Description	3291
5.597utility File Reference	3291
5.597.1 Detailed Description	3292
5.598valarray File Reference	3292
5.598.1 Detailed Description	3298

5.599valarray_after.h File Reference	3298
5.599.1 Detailed Description	3316
5.600valarray_array.h File Reference	3316
5.600.1 Detailed Description	3324
5.601valarray_array.tcc File Reference	3324
5.601.1 Detailed Description	3325
5.602valarray_before.h File Reference	3325
5.602.1 Detailed Description	3325
5.603vector File Reference	3325
5.603.1 Detailed Description	3325
5.604vector File Reference	3326
5.604.1 Detailed Description	3326
5.605vector File Reference	3326
5.605.1 Detailed Description	3327
5.606vector.tcc File Reference	3327
5.606.1 Detailed Description	3327
5.607vstring.h File Reference	3328
5.607.1 Detailed Description	3330
5.608vstring.tcc File Reference	3330
5.608.1 Detailed Description	3331
5.609vstring_fwd.h File Reference	3331
5.609.1 Detailed Description	3332
5.610vstring_util.h File Reference	3332
5.610.1 Detailed Description	3333
5.611workstealing.h File Reference	3333
5.611.1 Detailed Description	3333

Index

3333

1 Todo List

Member [__glibcxx_check_insert_range](#) ([_Position](#), [_First](#), [_Last](#))

We would like to be able to check for noninterference of [_Position](#) and the range [[_First](#), [_Last](#)), but that can't (in general) be done.

Member [__glibcxx_check_insert_range_after](#) ([_Position](#), [_First](#), [_Last](#))

We would like to be able to check for noninterference of [_Position](#) and the range [[_First](#), [_Last](#)), but that can't (in general) be done.

Member [__gnu_cxx::distance](#) ([_InputIterator](#) [__first](#), [_InputIterator](#) [__last](#), [_Distance](#) & [__n](#))

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_cxx::power` (`_Tp __x, _Integer __n, _MonoidOperation __monoid_op`)

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_cxx::power` (`_Tp __x, _Integer __n`)

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_cxx::random_sample` (`_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last`)

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_cxx::random_sample` (`_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand`)

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_cxx::random_sample_n` (`_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n`)

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_cxx::random_sample_n` (`_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand`)

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::rope<_CharT, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Class `__gnu_cxx::slist<_Tp, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator->()` const

Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Member `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator->()` const

Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Class `std::basic_string<_CharT, _Traits, _Alloc>`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::match_results<_Bi_iter, _Alloc>::format(_Out_iter __out, const char_type *__fmt_first, const char_type *__fmt_last, match_flag_type __flags=regex_constants::format_default)` const

Implement this function.

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator!=` (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator*` ()

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++` ()

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++` (int)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator->` ()

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator=` (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator==` (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator` ()

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default)`

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (const regex_iterator &__rhs)`

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Implement this function.

Member `std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Implement this function.

Implement this function.

Member `std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Implement this function.

Member `std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=` (const regex_token_iterator &__rhs)

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator*` ()

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++` (int)

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++` ()

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator->` ()

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator=` (const regex_token_iterator &__rhs)

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator==` (const regex_token_iterator &__rhs)

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator` (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector< int > &__submatches, regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator` ()

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator` (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[Nm], regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator` (const regex_token_iterator &__rhs)

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator` (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Member `std::regex_traits< _Ch_type >::lookup_classname` (_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const

Implement this function.

Member `std::regex_traits< _Ch_type >::lookup_collatename` (_Fwd_iter __first, _Fwd_iter __last) const

Implement this function.

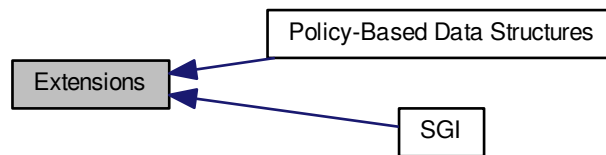
Member `std::regex_traits<_Ch_type>::transform_primary(_Fwd_iter __first, _Fwd_iter __last) const`

Implement this function.

2 Module Documentation

2.1 Extensions

Collaboration diagram for Extensions:



Modules

- [Policy-Based Data Structures](#)
- [SGI](#)

Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`
Template class __versa_string.
Data structure managing sequences of characters and character-like objects.

2.1.1 Detailed Description

Components generally useful that are not part of any standard.

2.2 SGI

Collaboration diagram for SGI:



Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
An SGI extension .
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
An SGI extension .
- struct `__gnu_cxx::constant_void_fun< _Result >`
An SGI extension .
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
An SGI extension .
- struct `__gnu_cxx::select2nd< _Pair >`
An SGI extension .
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`
An SGI extension .

Functions

- `template<typename _Tp >`
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `size_t std::Find_first () const noexcept`
- `size_t std::Find_next (size_t __prev) const noexcept`
- `template<class _Operation1, class _Operation2 >`
`unary_compose< _Operation1,`
`_Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`
`binary_compose< _Operation1,`
`_Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const`
`_Operation3 &__fn3)`
- `template<class _Result >`
`constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result,`
`_Result > __gnu_cxx::constant1 (const _Result &__val)`
- `template<class _Result >`
`constant_binary_fun< _Result,`
`_Result, _Result > __gnu_cxx::constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator,`
`_OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Distance >`
`void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
`__first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random-`
`AccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random-`
`AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`
`__out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`
`__out, const _Distance __n, _RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter`
`__result)`

- `bitset<_Nb> & std::_Unchecked_set (size_t __pos) noexcept`
- `bitset<_Nb> & std::_Unchecked_set (size_t __pos, int __val) noexcept`
- `bitset<_Nb> & std::_Unchecked_reset (size_t __pos) noexcept`
- `bitset<_Nb> & std::_Unchecked_flip (size_t __pos) noexcept`
- `constexpr bool std::_Unchecked_test (size_t __pos) const noexcept`

2.2.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function `compose2` takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

2.2.2 Function Documentation

2.2.2.1 `template<typename _Tp> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c)`

Find the median of three values.

Parameters

<code>__a</code>	A value.
<code>__b</code>	A value.
<code>__c</code>	A value.

Returns

One of `a`, `b` or `c`.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that $1 \leq m \leq n$ then the value returned will be `m`. This is an SGI extension.

Definition at line 546 of file `ext/algorithm`.

2.2.2.2 `template<typename _Tp, typename _Compare> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c, _Compare __comp)`

Find the median of three values using a predicate for comparison.

Parameters

<code>__a</code>	A value.
<code>__b</code>	A value.
<code>__c</code>	A value.
<code>__comp</code>	A binary predicate.

Returns

One of `a`, `b` or `c`.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that `comp (1, m)` and `comp (m, n)` are both true then the value returned will be `m`. This is an SGI extension.

Definition at line 580 of file `ext/algorithm`.

2.2.2.3 `size_t std::_Find_first () const [noexcept]`

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See Also

`_Find_next`

Definition at line 1354 of file `bitset`.

2.2.2.4 `size_t std::_Find_next (size_t __prev) const` `[noexcept]`

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

<code>__prev</code>	Where to start searching.
---------------------	---------------------------

See Also

`_Find_first`

Definition at line 1365 of file `bitset`.

2.2.2.5 `bitset<_Nb>& std::Unchecked_flip (size_t __pos)` `[noexcept]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1037 of file `bitset`.

Referenced by `std::flip()`.

2.2.2.6 `bitset<_Nb>& std::Unchecked_reset (size_t __pos)` `[noexcept]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1030 of file `bitset`.

Referenced by `std::reset()`.

2.2.2.7 `bitset<_Nb>& std::Unchecked_set (size_t __pos)` `[noexcept]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1013 of file `bitset`.

Referenced by `std::set()`.

2.2.2.8 `bitset<_Nb>& std::Unchecked_set (size_t __pos, int __val)` `[noexcept]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1020 of file `bitset`.

2.2.2.9 `constexpr bool std::Unchecked_test (size_t __pos) const` `[noexcept]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1044 of file `bitset`.

Referenced by `std::test()`.

2.2.2.10 `template<class _Operation1, class _Operation2> unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1 (const _Operation1 & __fn1, const _Operation2 & __fn2)` `[inline]`

An [SGI extension](#).

Definition at line 145 of file `ext/functional`.

2.2.2.11 `template<class _Operation1 , class _Operation2 , class _Operation3 > binary_compose<_Operation1, _Operation2, _Operation3> __gnu_cxx::compose2 (const _Operation1 & __fn1, const _Operation2 & __fn2, const _Operation3 & __fn3)`
`[inline]`

An [SGI extension](#) .

Definition at line 172 of file ext/functional.

2.2.2.12 `template<class _Result > constant_void_fun<_Result> __gnu_cxx::constant0 (const _Result & __val)` `[inline]`

An [SGI extension](#) .

Definition at line 330 of file ext/functional.

2.2.2.13 `template<class _Result > constant_unary_fun<_Result, _Result> __gnu_cxx::constant1 (const _Result & __val)`
`[inline]`

An [SGI extension](#) .

Definition at line 336 of file ext/functional.

2.2.2.14 `template<class _Result > constant_binary_fun<_Result,_Result,_Result> __gnu_cxx::constant2 (const _Result & __val)`
`[inline]`

An [SGI extension](#) .

Definition at line 342 of file ext/functional.

2.2.2.15 `template<typename _InputIterator , typename _Size , typename _OutputIterator > pair<_InputIterator, _OutputIterator> __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)` `[inline]`

Copies the range [first,first+count) into [result,result+count).

Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

A std::pair composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 120 of file ext/algorithm.

References `std::__iterator_category()`.

2.2.2.16 `template<typename _InputIterator , typename _Distance > void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance & __n)` `[inline]`

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Definition at line 105 of file ext/iterator.

References `std::__iterator_category()`.

2.2.2.17 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::plus< _Tp >) [inline]`

An [SGI extension](#) .

Definition at line 87 of file ext/functional.

2.2.2.18 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >) [inline]`

An [SGI extension](#) .

Definition at line 93 of file ext/functional.

2.2.2.19 `template<typename _InputIterator1 , typename _InputIterator2 > int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

`memcmp` on steroids.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

An int, as with `memcmp`.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 201 of file ext/algorithm.

2.2.2.20 `template<typename _Tp , typename _Integer , typename _MonoidOperation > _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op) [inline]`

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

Definition at line 113 of file ext/numeric.

2.2.2.21 `template<typename _Tp , typename _Integer > _Tp __gnu_cxx::power (_Tp __x, _Integer __n) [inline]`

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

Definition at line 123 of file ext/numeric.

```
2.2.2.22 template<typename _InputIterator , typename _RandomAccessIterator > _RandomAccessIterator
    __gnu_cxx::random_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first,
        _RandomAccessIterator __out_last ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

Definition at line 388 of file ext/algorithm.

```
2.2.2.23 template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator >
    _RandomAccessIterator __gnu_cxx::random_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator
        __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator & __rand ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

Definition at line 411 of file ext/algorithm.

```
2.2.2.24 template<typename _ForwardIterator , typename _OutputIterator , typename _Distance > _OutputIterator
    __gnu_cxx::random_sample_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n )
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

Definition at line 267 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.25 template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename
    _RandomNumberGenerator > _OutputIterator __gnu_cxx::random_sample_n ( _ForwardIterator __first, _ForwardIterator
        __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator & __rand )
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

Definition at line 301 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.26 template<typename _InputIter , typename _Size , typename _ForwardIter > pair<_InputIter, _ForwardIter>
    __gnu_cxx::uninitialized_copy_n ( _InputIter __first, _Size __count, _ForwardIter __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	Length
<code>__result</code>	An output iterator.

Returns

`__result + (__first + __count)`

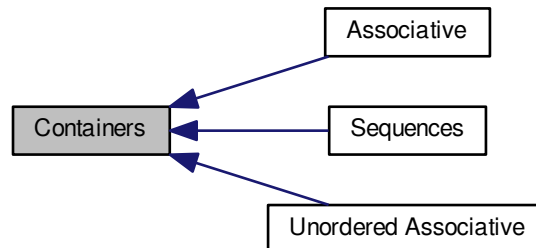
Like `copy()`, but does not require an initialized output range.

Definition at line 122 of file `ext/memory`.

References `std::__iterator_category()`.

2.3 Containers

Collaboration diagram for Containers:



Modules

- [Associative](#)
- [Sequences](#)
- [Unordered Associative](#)

Classes

- class `std::tr2::dynamic_bitset<_WordT, _Alloc>`
The `dynamic_bitset` class represents a sequence of bits.
- class `std::tr2::dynamic_bitset<_WordT, _Alloc>::reference`

2.3.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

2.4 Sequences

Collaboration diagram for Sequences:



Classes

- struct `std::array<_Tp, _Nm>`
A standard container for storing a fixed size sequence of elements.
- class `std::basic_string<_CharT, _Traits, _Alloc>`
Managing sequences of characters and character-like objects.
- class `std::deque<_Tp, _Alloc>`
A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.
- class `std::forward_list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class `std::list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class `std::priority_queue<_Tp, _Sequence, _Compare>`
A standard container automatically sorting its contents.
- class `std::queue<_Tp, _Sequence>`
A standard container giving FIFO behavior.
- class `std::stack<_Tp, _Sequence>`
A standard container giving FILO behavior.
- class `std::vector<_Tp, _Alloc>`
A standard container which offers fixed time access to individual elements in any order.
- class `std::vector<bool, _Alloc>`
A specialization of vector for booleans which offers fixed time access to individual elements in any order.

2.4.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence,

`list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

2.5 Associative

Collaboration diagram for Associative:



Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multiset< _Key, _Compare, _Alloc >`
A standard container made up of elements, which can be retrieved in logarithmic time.
- class `std::set< _Key, _Compare, _Alloc >`
A standard container made up of unique keys, which can be retrieved in logarithmic time.

2.5.1 Detailed Description

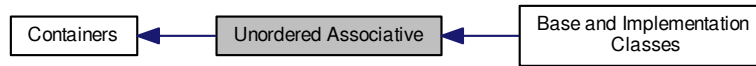
Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

2.6 Unordered Associative

Collaboration diagram for Unordered Associative:



Modules

- [Base and Implementation Classes](#)

Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

2.6.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

2.7 Diagnostics

Collaboration diagram for Diagnostics:



Modules

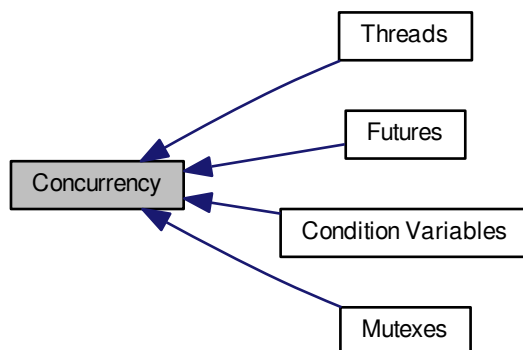
- [Exceptions](#)

2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.8 Concurrency

Collaboration diagram for Concurrency:



Modules

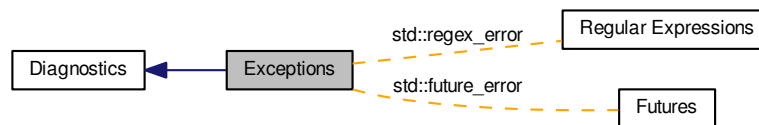
- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

2.8.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

2.9 Exceptions

Collaboration diagram for Exceptions:



Classes

- class `__cxxabiv1::__forced_unwind`
*Thrown as part of forced unwinding.
 A magic placeholder class that can be caught by reference to recognize forced unwinding.*
- struct `__gnu_cxx::forced_error`
Thrown by exception safety machinery.
- class `__gnu_cxx::recursive_init_error`
*Exception thrown by `__cxa_guard_acquire`.
 6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `std::__exception_ptr::exception_ptr`
An opaque pointer to an arbitrary exception.
- class `std::bad_alloc`
*Exception possibly thrown by `new`.
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*
- class `std::bad_cast`
*Thrown during incorrect typecasting.
 If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class `std::bad_exception`
- class `std::bad_function_call`
Exception class thrown when class template function's `operator()` is called with an empty target.
- class `std::bad_typeid`
Thrown when a `NULL` pointer in a `typeid` expression is used.
- class `std::bad_weak_ptr`
Exception possibly thrown by `shared_ptr`.
- class `std::domain_error`
- class `std::exception`
Base class for all library exceptions.
- class `std::future_error`
Exception type thrown by futures.
- class `std::invalid_argument`
- class `std::ios_base::failure`

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

- class `std::length_error`
- class `std::logic_error`

One of two subclasses of exception.

- class `std::nested_exception`

Exception class with `exception_ptr` data member.

- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `std::regex_error`

A regular expression exception class.

The regular expression library throws objects of this class on error.

- class `std::runtime_error`

One of two subclasses of exception.

- class `std::system_error`

Thrown to indicate error code of underlying system.

- class `std::underflow_error`

Typedefs

- typedef void(* `std::terminate_handler`)()
- typedef void(* `std::unexpected_handler`)()

Functions

- template<typename `_Ex` >
const nested_exception * **std::__get_nested_exception** (const `_Ex` &__ex)
- template<typename `_Ex` >
void **std::__throw_with_nested** (`_Ex` &&, const nested_exception * = 0) __attribute__((__noreturn__))
- template<typename `_Ex` >
void **std::__throw_with_nested** (`_Ex` &&, ...) __attribute__((__noreturn__))
- void `__gnu_cxx::__verbose_terminate_handler` ()
- template<typename `_Ex` >
exception_ptr **std::copy_exception** (`_Ex` __ex) noexcept
- exception_ptr **std::current_exception** () noexcept
- template<typename `_Ex` >
exception_ptr **std::make_exception_ptr** (`_Ex` __ex) noexcept
- void **std::rethrow_exception** (exception_ptr) __attribute__((__noreturn__))
- template<typename `_Ex` >
void **std::rethrow_if_nested** (const `_Ex` &__ex)
- void **std::rethrow_if_nested** (const nested_exception &__ex)
- terminate_handler **std::set_terminate** (terminate_handler) noexcept
- unexpected_handler **std::set_unexpected** (unexpected_handler) noexcept
- void **std::terminate** () noexcept __attribute__((__noreturn__))
- template<typename `_Ex` >
void **std::throw_with_nested** (`_Ex` __ex)
- bool **std::uncaught_exception** () noexcept __attribute__((__pure__))
- void **std::unexpected** () __attribute__((__noreturn__))

2.9.1 Detailed Description

Classes and functions for reporting errors via exception classes.

2.9.2 Typedef Documentation

2.9.2.1 `typedef void(* std::terminate_handler)()`

If you write a replacement terminate handler, it must be of this type.

Definition at line 87 of file `exception`.

2.9.2.2 `typedef void(* std::unexpected_handler)()`

If you write a replacement unexpected handler, it must be of this type.

Definition at line 90 of file `exception`.

2.9.3 Function Documentation

2.9.3.1 `void __gnu_cxx::__verbose_terminate_handler ()`

A replacement for the standard `terminate_handler` which prints more information about the terminating exception (if any) on `stderr`.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler  
)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.-html>

In 3.4 and later, this is on by default.

2.9.3.2 `template<typename _Ex > exception_ptr std::copy_exception (_Ex __ex) [noexcept]`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 167 of file `exception_ptr.h`.

References `std::current_exception()`.

2.9.3.3 `exception_ptr std::current_exception () [noexcept]`

Obtain an `exception_ptr` to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by `std::copy_exception()`.

2.9.3.4 `template<typename _Ex > exception_ptr std::make_exception_ptr (_Ex __ex) [noexcept]`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 186 of file `exception_ptr.h`.

2.9.3.5 `void std::rethrow_exception (exception_ptr)`

Throw the object pointed to by the `exception_ptr`.

2.9.3.6 `template<typename _Ex> void std::rethrow_if_nested (const _Ex & __ex) [inline]`

If `__ex` is derived from `nested_exception`, `__ex.rethrow_nested()`.

Definition at line 146 of file `nested_exception.h`.

2.9.3.7 `void std::rethrow_if_nested (const nested_exception & __ex) [inline]`

Overload, See N2619.

Definition at line 154 of file `nested_exception.h`.

2.9.3.8 `terminate_handler std::set_terminate (terminate_handler) [noexcept]`

Takes a new handler function as an argument, returns the old function.

2.9.3.9 `unexpected_handler std::set_unexpected (unexpected_handler) [noexcept]`

Takes a new handler function as an argument, returns the old function.

2.9.3.10 `void std::terminate () [noexcept]`

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

2.9.3.11 `template<typename _Ex> void std::throw_with_nested (_Ex __ex) [inline]`

If `__ex` is derived from `nested_exception`, `__ex`. Else, an implementation-defined object derived from both.

Definition at line 136 of file `nested_exception.h`.

2.9.3.12 `bool std::uncaught_exception () [noexcept]`

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).'

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::~~sentry()`.

2.9.3.13 `void std::unexpected ()`

The runtime will call this function if an exception is thrown which violates the function's exception specification.

2.10 Time

Collaboration diagram for Time:



Namespaces

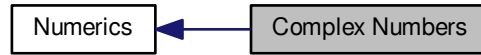
- namespace [std::chrono](#)

2.10.1 Detailed Description

Classes and functions for time.

2.11 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- struct `std::complex< double >`
26.2.3 complex specializations complex<double> specialization
- struct `std::complex< float >`
26.2.3 complex specializations complex<float> specialization
- struct `std::complex< long double >`
26.2.3 complex specializations complex<long double> specialization

Functions

- constexpr `std::complex< float >::complex` (const complex< long double > &)
- template<typename `_Tp` >
`_Tp std::__complex_abs` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`std::complex< _Tp > std::tr1::__complex_acosh` (const `std::complex< _Tp >` &`_z`)
- template<typename `_Tp` >
`_Tp std::__complex_arg` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`std::complex< _Tp > std::tr1::__complex_asinh` (const `std::complex< _Tp >` &`_z`)
- template<typename `_Tp` >
`std::complex< _Tp > std::tr1::__complex_atanh` (const `std::complex< _Tp >` &`_z`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_cos` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_cosh` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_exp` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_log` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_pow` (const complex< `_Tp` > &`_x`, const complex< `_Tp` > &`_y`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_sin` (const complex< `_Tp` > &`_z`)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_sinh` (const complex< `_Tp` > &`_z`)

- `template<typename _Tp >`
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< typename`
`__gnu_cxx::__promote< _Tp >`
`::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::norm (const complex< _Tp > &)`
- `complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &)`
- `template<typename _Up >`
`complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &)`
- `complex< _Tp > & std::complex< _Tp >::operator/= (const _Tp &)`

- `template<typename _Up >`
`complex< _Tp > & std::complex< _Tp >::operator/= (const complex< _Up > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `complex< _Tp > & std::complex< _Tp >::operator= (const _Tp &)`
- `template<typename _Up >`
`complex< _Tp > & std::complex< _Tp >::operator= (const complex< _Up > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename`
`__gnu_cxx::__promote_2< _Tp,`
`_Up >::__type > std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename`
`__gnu_cxx::__promote_2< _Tp,`
`_Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename`
`__gnu_cxx::__promote_2< _Tp,`
`_Up >::__type > std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename`
`__gnu_cxx::__promote_2< _Tp,`
`_Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tanh (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

2.11.1 Detailed Description

Classes and functions for complex numbers.

2.11.2 Function Documentation

2.11.2.1 `template<typename _Tp > _Tp std::abs (const complex< _Tp > &__z) [inline]`

Return magnitude of `z`.

Definition at line 599 of file complex.

Referenced by `std::tr1::fabs()`, `std::fabs()`, `std::binomial_distribution<_IntType>::operator()()`, and `std::poisson_distribution<_IntType>::operator()()`.

2.11.2.2 `template<typename _Tp> std::complex<_Tp> std::tr1::acosh (const std::complex<_Tp> & __z)`
[inline]

`acosh(__z)` [8.1.5].

Definition at line 215 of file `tr1/complex`.

2.11.2.3 `template<typename _Tp> _Tp std::arg (const complex<_Tp> & __z)` [inline]

Return phase angle of `z`.

Definition at line 626 of file `complex`.

Referenced by `std::arg()`.

2.11.2.4 `template<typename _Tp> std::complex<_Tp> std::tr1::asinh (const std::complex<_Tp> & __z)` [inline]

`asinh(__z)` [8.1.6].

Definition at line 254 of file `tr1/complex`.

2.11.2.5 `template<typename _Tp> std::complex<_Tp> std::tr1::atanh (const std::complex<_Tp> & __z)` [inline]

`atanh(__z)` [8.1.7].

Definition at line 298 of file `tr1/complex`.

2.11.2.6 `template<typename _Tp> complex<_Tp> std::conj (const complex<_Tp> & __z)` [inline]

Return complex conjugate of `z`.

Definition at line 672 of file `complex`.

2.11.2.7 `template<typename _Tp> complex<_Tp> std::cos (const complex<_Tp> & __z)` [inline]

Return complex cosine of `z`.

Definition at line 704 of file `complex`.

Referenced by `std::polar()`.

2.11.2.8 `template<typename _Tp> complex<_Tp> std::cosh (const complex<_Tp> & __z)` [inline]

Return complex hyperbolic cosine of `z`.

Definition at line 734 of file `complex`.

2.11.2.9 `template<typename _Tp> complex<_Tp> std::exp (const complex<_Tp> & __z)` [inline]

Return complex base `e` exponential of `z`.

Definition at line 760 of file `complex`.

Referenced by `std::pow()`.

2.11.2.10 `template<typename _Tp> std::complex<_Tp> std::tr1::fabs (const std::complex<_Tp> & __z)` [inline]

`fabs(__z)` [8.1.8].

Definition at line 307 of file tr1/complex.

References `std::abs()`.

2.11.2.11 `template<typename _Tp> complex<_Tp> std::log (const complex<_Tp> & __z) [inline]`

Return complex natural logarithm of z .

Definition at line 787 of file complex.

Referenced by `std::generate_canonical()`, `std::log10()`, `std::normal_distribution<_RealType>::operator()`, `std::gamma_distribution<_RealType>::operator()`, `std::binomial_distribution<_IntType>::operator()`, `std::poisson_distribution<_IntType>::operator()`, and `std::pow()`.

2.11.2.12 `template<typename _Tp> complex<_Tp> std::log10 (const complex<_Tp> & __z) [inline]`

Return complex base 10 logarithm of z .

Definition at line 792 of file complex.

References `std::log()`.

2.11.2.13 `template<typename _Tp> _Tp std::norm (const complex<_Tp> & __z) [inline]`

Return z magnitude squared.

Definition at line 659 of file complex.

Referenced by `std::complex<_Tp>::operator/()`.

2.11.2.14 `template<typename _Tp> constexpr bool std::operator!= (const complex<_Tp> & __x, const complex<_Tp> & __y) [inline]`

Return false if x is equal to y .

Definition at line 474 of file complex.

2.11.2.15 `template<typename _Tp> constexpr bool std::operator!= (const complex<_Tp> & __x, const _Tp & __y) [inline]`

Return false if x is equal to y .

Definition at line 479 of file complex.

2.11.2.16 `template<typename _Tp> constexpr bool std::operator!= (const _Tp & __x, const complex<_Tp> & __y) [inline]`

Return false if x is equal to y .

Definition at line 484 of file complex.

2.11.2.17 `template<typename _Tp> complex<_Tp> std::operator* (const complex<_Tp> & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x times y .

Definition at line 384 of file complex.

2.11.2.18 `template<typename _Tp> complex<_Tp> std::operator* (const complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x times y .

Definition at line 393 of file complex.

2.11.2.19 `template<typename _Tp> complex<_Tp> std::operator* (const _Tp & __x, const complex<_Tp> & __y)
[inline]`

Return new complex value x times y .

Definition at line 402 of file complex.

2.11.2.20 `template<typename _Tp> complex<_Tp> & complex<_Tp>::operator*= (const _Tp & __t)`

Multiply this complex number by t .

Definition at line 243 of file complex.

2.11.2.21 `template<typename _Tp> template<typename _Up> complex<_Tp> & complex<_Tp>::operator*= (const
complex<_Up> & __z)`

Multiply this complex number by z .

Definition at line 297 of file complex.

2.11.2.22 `template<typename _Tp> complex<_Tp> std::operator+ (const complex<_Tp> & __x, const complex<_Tp> & __y
) [inline]`

Return new complex value x plus y .

Definition at line 324 of file complex.

2.11.2.23 `template<typename _Tp> complex<_Tp> std::operator+ (const complex<_Tp> & __x, const _Tp & __y)
[inline]`

Return new complex value x plus y .

Definition at line 333 of file complex.

2.11.2.24 `template<typename _Tp> complex<_Tp> std::operator+ (const _Tp & __x, const complex<_Tp> & __y)
[inline]`

Return new complex value x plus y .

Definition at line 342 of file complex.

2.11.2.25 `template<typename _Tp> complex<_Tp> std::operator+ (const complex<_Tp> & __x) [inline]`

Return x .

Definition at line 443 of file complex.

2.11.2.26 `template<typename _Tp> template<typename _Up> complex<_Tp> & complex<_Tp>::operator+= (const
complex<_Up> & __z)`

Add z to this complex number.

Definition at line 274 of file complex.

2.11.2.27 `template<typename _Tp> complex<_Tp> std::operator- (const complex<_Tp> & __x, const complex<_Tp> & __y
) [inline]`

Return new complex value x minus y .

Definition at line 354 of file complex.

2.11.2.28 `template<typename _Tp> complex<_Tp> std::operator- (const complex< _Tp> & __x, const _Tp & __y)`
`[inline]`

Return new complex value x minus y .

Definition at line 363 of file complex.

2.11.2.29 `template<typename _Tp> complex<_Tp> std::operator- (const _Tp & __x, const complex< _Tp> & __y)`
`[inline]`

Return new complex value x minus y .

Definition at line 372 of file complex.

2.11.2.30 `template<typename _Tp> complex<_Tp> std::operator- (const complex< _Tp> & __x)` `[inline]`

Return complex negation of x .

Definition at line 449 of file complex.

2.11.2.31 `template<typename _Tp> template<typename _Up> complex< _Tp> & complex< _Tp>::operator-= (const complex< _Up> & __z)`

Subtract z from this complex number.

Definition at line 285 of file complex.

2.11.2.32 `template<typename _Tp> complex<_Tp> std::operator/ (const complex< _Tp> & __x, const complex< _Tp> & __y`
`)` `[inline]`

Return new complex value x divided by y .

Definition at line 414 of file complex.

2.11.2.33 `template<typename _Tp> complex<_Tp> std::operator/ (const complex< _Tp> & __x, const _Tp & __y)`
`[inline]`

Return new complex value x divided by y .

Definition at line 423 of file complex.

2.11.2.34 `template<typename _Tp> complex<_Tp> std::operator/ (const _Tp & __x, const complex< _Tp> & __y)`
`[inline]`

Return new complex value x divided by y .

Definition at line 432 of file complex.

2.11.2.35 `template<typename _Tp> complex< _Tp> & complex< _Tp>::operator/= (const _Tp & __t)`

Divide this complex number by t .

Definition at line 253 of file complex.

2.11.2.36 `template<typename _Tp> template<typename _Up> complex< _Tp> & complex< _Tp>::operator/= (const complex< _Up> & __z)`

Divide this complex number by z .

Definition at line 310 of file complex.

References `std::norm()`.

2.11.2.37 `template<typename _Tp, typename _CharT, class _Traits> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> & __os, const complex<_Tp> & __x)`

Insertion operator for complex values.

Definition at line 524 of file complex.

References `std::ios_base::flags()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::ios_base::precision()`, and `std::basic_ostringstream<_CharT, _Traits, _Alloc>::str()`.

2.11.2.38 `template<typename _Tp> complex<_Tp> & complex<_Tp>::operator= (const _Tp & __t)`

Assign this complex number to scalar *t*.

Definition at line 233 of file complex.

2.11.2.39 `template<typename _Tp> template<typename _Up> complex<_Tp> & complex<_Tp>::operator= (const complex<_Up> & __z)`

Assign this complex number to complex *z*.

Definition at line 263 of file complex.

2.11.2.40 `template<typename _Tp> constexpr bool std::operator== (const complex<_Tp> & __x, const complex<_Tp> & __y) [inline]`

Return true if *x* is equal to *y*.

Definition at line 456 of file complex.

2.11.2.41 `template<typename _Tp> constexpr bool std::operator== (const complex<_Tp> & __x, const _Tp & __y) [inline]`

Return true if *x* is equal to *y*.

Definition at line 461 of file complex.

2.11.2.42 `template<typename _Tp> constexpr bool std::operator== (const _Tp & __x, const complex<_Tp> & __y) [inline]`

Return true if *x* is equal to *y*.

Definition at line 466 of file complex.

2.11.2.43 `template<typename _Tp, typename _CharT, class _Traits> basic_istream<_CharT, _Traits>& std::operator>> (basic_istream<_CharT, _Traits> & __is, complex<_Tp> & __x)`

Extraction operator for complex values.

Definition at line 491 of file complex.

References `std::ios_base::failbit`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

2.11.2.44 `template<typename _Tp> complex<_Tp> std::polar (const _Tp & __rho, const _Tp & __theta = 0) [inline]`

Return complex with magnitude *rho* and angle *theta*.

Definition at line 667 of file complex.

References `std::cos()`, and `std::sin()`.

Referenced by `std::pow()`.

2.11.2.45 `template<typename _Tp> complex< _Tp> std::pow (const complex< _Tp> & __x, const _Tp & __y)`

Return x to the y 'th power.

Definition at line 989 of file complex.

References `std::exp()`, `std::log()`, and `std::polar()`.

Referenced by `std::gamma_distribution< _RealType>::operator()()`, `std::pow()`, and `std::tr1::pow()`.

2.11.2.46 `template<typename _Tp> complex< _Tp> std::pow (const complex< _Tp> & __x, const complex< _Tp> & __y)`
`[inline]`

Return x to the y 'th power.

Definition at line 1028 of file complex.

2.11.2.47 `template<typename _Tp> complex< _Tp> std::pow (const _Tp & __x, const complex< _Tp> & __y)` `[inline]`

Return x to the y 'th power.

Definition at line 1034 of file complex.

References `std::log()`, `std::polar()`, and `std::pow()`.

2.11.2.48 `template<typename _Tp, typename _Up> std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type>`
`std::tr1::pow (const std::complex< _Tp> & __x, const _Up & __y)` `[inline]`

Additional overloads [8.1.9].

Definition at line 348 of file tr1/complex.

References `std::pow()`.

2.11.2.49 `template<typename _Tp> complex< _Tp> std::sin (const complex< _Tp> & __z)` `[inline]`

Return complex sine of z .

Definition at line 822 of file complex.

Referenced by `std::polar()`.

2.11.2.50 `template<typename _Tp> complex< _Tp> std::sinh (const complex< _Tp> & __z)` `[inline]`

Return complex hyperbolic sine of z .

Definition at line 852 of file complex.

2.11.2.51 `template<typename _Tp> complex< _Tp> std::sqrt (const complex< _Tp> & __z)` `[inline]`

Return complex square root of z .

Definition at line 896 of file complex.

Referenced by `std::normal_distribution< _RealType>::operator()()`, and `std::student_t_distribution< _RealType>::operator()()`.

2.11.2.52 `template<typename _Tp> complex< _Tp > std::tan (const complex< _Tp > & _z) [inline]`

Return complex tangent of z .

Definition at line 923 of file `complex`.

2.11.2.53 `template<typename _Tp> complex< _Tp > std::tanh (const complex< _Tp > & _z) [inline]`

Return complex hyperbolic tangent of z .

Definition at line 951 of file `complex`.

2.12 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class `std::condition_variable`
condition_variable
- class `std::condition_variable_any`
condition_variable_any

Enumerations

- enum `std::cv_status`

2.12.1 Detailed Description

Classes for `condition_variable` support.

2.12.2 Enumeration Type Documentation

2.12.2.1 enum `std::cv_status`

`cv_status`

Definition at line 56 of file `condition_variable`.

2.13 Futures

Collaboration diagram for Futures:



Classes

- class `std::future_error`
Exception type thrown by futures.
- struct `std::is_error_code_enum< future_errc >`
Specialization.

Enumerations

- enum `std::future_errc`
- enum `std::future_status`
- enum `std::launch`

Functions

- template<typename `_Fn` , typename... `_Args`>
future< typename `result_of`
< `_Fn`(`_Args...`)>::type > `std::async` (`launch` `__policy`, `_Fn` && `__fn`, `_Args` &&...`__args`)
- template<typename `_Fn` , typename... `_Args`>
future< typename `result_of`
< `_Fn`(`_Args...`)>::type > `std::async` (`_Fn` && `__fn`, `_Args` &&...`__args`)
- const error_category & `std::future_category` () noexcept
- error_code `std::make_error_code` (`future_errc` `__errc`) noexcept
- error_condition `std::make_error_condition` (`future_errc` `__errc`) noexcept
- constexpr launch `std::operator&` (`launch` `__x`, `launch` `__y`)
- launch & `std::operator&=` (`launch` & `__x`, `launch` `__y`)
- constexpr launch `std::operator^` (`launch` `__x`, `launch` `__y`)
- launch & `std::operator^=` (`launch` & `__x`, `launch` `__y`)
- constexpr launch `std::operator|` (`launch` `__x`, `launch` `__y`)
- launch & `std::operator|=` (`launch` & `__x`, `launch` `__y`)
- constexpr launch `std::operator~` (`launch` `__x`)

2.13.1 Detailed Description

Classes for futures support.

2.13.2 Enumeration Type Documentation

2.13.2.1 `enum std::future_errc`

Error code for futures.

Definition at line 63 of file future.

2.13.2.2 `enum std::future_status`

Status code for futures.

Definition at line 162 of file future.

2.13.2.3 `enum std::launch`

Launch code for futures.

Definition at line 125 of file future.

2.13.3 Function Documentation

2.13.3.1 `const error_category& std::future_category () [noexcept]`

Points to a statically-allocated object derived from `error_category`.

Referenced by `std::make_error_code()`, and `std::make_error_condition()`.

2.13.3.2 `error_code std::make_error_code (future_errc __errc) [inline],[noexcept]`

Overload for `make_error_code`.

Definition at line 81 of file future.

References `std::future_category()`.

2.13.3.3 `error_condition std::make_error_condition (future_errc __errc) [inline],[noexcept]`

Overload for `make_error_condition`.

Definition at line 86 of file future.

References `std::future_category()`.

2.14 I/O

Classes

- class [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#)
*Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#)
*Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class [std::basic_filebuf< _CharT, _Traits >](#)
The actual work of input and output (for files).
- class [std::basic_fstream< _CharT, _Traits >](#)
Controlling input and output for files.
- class [std::basic_ifstream< _CharT, _Traits >](#)
Controlling input for files.
- class [std::basic_ofstream< _CharT, _Traits >](#)
Controlling output for files.
- class [std::ios_base](#)
*The base of the I/O class hierarchy.
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the `openmodes`).*

Typedefs

- typedef `basic_filebuf< char >` [std::filebuf](#)
- typedef `basic_fstream< char >` [std::fstream](#)
- typedef `basic_ifstream< char >` [std::ifstream](#)
- typedef `basic_ios< char >` [std::ios](#)
- typedef `basic_iostream< char >` [std::iostream](#)
- typedef `basic_istream< char >` [std::istream](#)
- typedef `basic_istreamstream< char >` [std::istreamstream](#)
- typedef `basic_ofstream< char >` [std::ofstream](#)
- typedef `basic_ostream< char >` [std::ostream](#)
- typedef `basic_ostreamstream< char >` [std::ostreamstream](#)
- typedef `basic_streambuf< char >` [std::streambuf](#)
- typedef `basic_stringbuf< char >` [std::stringbuf](#)
- typedef `basic_stringstream< char >` [std::stringstream](#)
- typedef `basic_filebuf< wchar_t >` [std::wfilebuf](#)
- typedef `basic_fstream< wchar_t >` [std::wfstream](#)
- typedef `basic_ifstream< wchar_t >` [std::wifstream](#)
- typedef `basic_ios< wchar_t >` [std::wios](#)
- typedef `basic_iostream< wchar_t >` [std::wiostream](#)
- typedef `basic_istream< wchar_t >` [std::wistream](#)
- typedef `basic_istreamstream`
`< wchar_t >` [std::wistreamstream](#)
- typedef `basic_ofstream< wchar_t >` [std::wofstream](#)
- typedef `basic_ostream< wchar_t >` [std::wostream](#)

- `typedef basic_ostringstream`
 `< wchar_t > std::wostringstream`
- `typedef basic_streambuf< wchar_t > std::wstreambuf`
- `typedef basic_stringbuf< wchar_t > std::wstringbuf`
- `typedef basic_stringstream`
 `< wchar_t > std::wstringstream`

2.14.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.-html>

2.14.2 Typedef Documentation

2.14.2.1 `typedef basic_filebuf<char> std::filebuf`

Class for `char` file buffers.

Definition at line 154 of file `iosfwd`.

2.14.2.2 `typedef basic_fstream<char> std::fstream`

Class for `char` mixed input and output file streams.

Definition at line 163 of file `iosfwd`.

2.14.2.3 `typedef basic_ifstream<char> std::ifstream`

Class for `char` input file streams.

Definition at line 157 of file `iosfwd`.

2.14.2.4 `typedef basic_ios<char> std::ios`

Base class for `char` streams.

Definition at line 123 of file `iosfwd`.

2.14.2.5 `typedef basic_iostream<char> std::iostream`

Base class for `char` mixed input and output streams.

Definition at line 139 of file `iosfwd`.

2.14.2.6 typedef basic_istream<char> std::istream

Base class for `char` input streams.

Definition at line 133 of file `iosfwd`.

2.14.2.7 typedef basic_istream<char> std::istream

Class for `char` input memory streams.

Definition at line 145 of file `iosfwd`.

2.14.2.8 typedef basic_ofstream<char> std::ofstream

Class for `char` output file streams.

Definition at line 160 of file `iosfwd`.

2.14.2.9 typedef basic_ostream<char> std::ostream

Base class for `char` output streams.

Definition at line 136 of file `iosfwd`.

2.14.2.10 typedef basic_ostream<char> std::ostream

Class for `char` output memory streams.

Definition at line 148 of file `iosfwd`.

2.14.2.11 typedef basic_streambuf<char> std::streambuf

Base class for `char` buffers.

Definition at line 130 of file `iosfwd`.

2.14.2.12 typedef basic_stringbuf<char> std::stringbuf

Class for `char` memory buffers.

Definition at line 142 of file `iosfwd`.

2.14.2.13 typedef basic_stringstream<char> std::stringstream

Class for `char` mixed input and output memory streams.

Definition at line 151 of file `iosfwd`.

2.14.2.14 typedef basic_filebuf<wchar_t> std::wfilebuf

Class for `wchar_t` file buffers.

Definition at line 194 of file `iosfwd`.

2.14.2.15 typedef basic_fstream<wchar_t> std::wfstream

Class for `wchar_t` mixed input and output file streams.

Definition at line 203 of file `iosfwd`.

2.14.2.16 typedef basic_ifstream<wchar_t> std::wifstream

Class for `wchar_t` input file streams.

Definition at line 197 of file `iosfwd`.

2.14.2.17 typedef basic_ios<wchar_t> std::wios

Base class for `wchar_t` streams.

Definition at line 167 of file `iosfwd`.

2.14.2.18 typedef basic_iostream<wchar_t> std::wiostream

Base class for `wchar_t` mixed input and output streams.

Definition at line 179 of file `iosfwd`.

2.14.2.19 typedef basic_istream<wchar_t> std::wistream

Base class for `wchar_t` input streams.

Definition at line 173 of file `iosfwd`.

2.14.2.20 typedef basic_istream<wchar_t> std::wistream

Class for `wchar_t` input memory streams.

Definition at line 185 of file `iosfwd`.

2.14.2.21 typedef basic_ofstream<wchar_t> std::wofstream

Class for `wchar_t` output file streams.

Definition at line 200 of file `iosfwd`.

2.14.2.22 typedef basic_ostream<wchar_t> std::wostream

Base class for `wchar_t` output streams.

Definition at line 176 of file `iosfwd`.

2.14.2.23 typedef basic_ostream<wchar_t> std::wostream

Class for `wchar_t` output memory streams.

Definition at line 188 of file `iosfwd`.

2.14.2.24 typedef basic_streambuf<wchar_t> std::wstreambuf

Base class for `wchar_t` buffers.

Definition at line 170 of file `iosfwd`.

2.14.2.25 typedef basic_stringbuf<wchar_t> std::wstringbuf

Class for `wchar_t` memory buffers.

Definition at line 182 of file `iosfwd`.

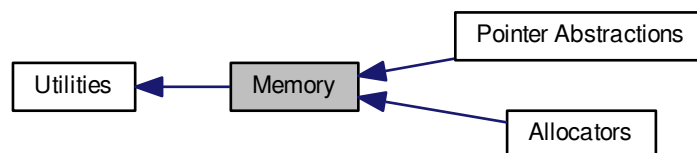
2.14.2.26 `typedef basic_stringstream<wchar_t> std::wstringstream`

Class for `wchar_t` mixed input and output memory streams.

Definition at line 191 of file `iosfwd`.

2.15 Memory

Collaboration diagram for Memory:



Modules

- [Allocators](#)
- [Pointer Abstractions](#)

2.15.1 Detailed Description

Components for memory allocation, deallocation, and management.

2.16 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- struct `owner_less< _Tp >`
Primary template owner_less.
- struct `std::default_delete< _Tp >`
Primary template, default_delete.
- struct `std::default_delete< _Tp[] >`
Specialization, default_delete.
- class `std::enable_shared_from_this< _Tp >`
Base class allowing use of member function shared_from_this.
- struct `std::hash< shared_ptr< _Tp > >`
std::hash specialization for shared_ptr.
- struct `std::hash< unique_ptr< _Tp, _Dp > >`
std::hash specialization for unique_ptr.
- struct `std::owner_less< shared_ptr< _Tp > >`
Partial specialization of owner_less for shared_ptr.
- struct `std::owner_less< weak_ptr< _Tp > >`
Partial specialization of owner_less for weak_ptr.
- struct `std::pointer_traits< _Ptr >`
Uniform interface to all pointer-like types.
- struct `std::pointer_traits< _Tp * >`
Partial specialization for built-in pointers.
- class `std::shared_ptr< _Tp >`
A smart pointer with reference-counted copy semantics.
- class `std::unique_ptr< _Tp, _Dp >`
20.7.1.2 unique_ptr for single objects.
- class `std::unique_ptr< _Tp[], _Dp >`
20.7.1.3 unique_ptr for array objects with a runtime length
- class `std::weak_ptr< _Tp >`
A smart pointer with weak semantics.

Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr< _Tp > std::allocate_shared (const _Alloc &__a, _Args &&...__args)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > std::make_shared (_Args &&...__args)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`

- `template<typename _Tp >`
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Dp >`
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

2.16.1 Detailed Description

Smart pointers, etc.

2.16.2 Function Documentation

2.16.2.1 `template<typename _Tp, typename _Alloc, typename... _Args> shared_ptr<_Tp> std::allocate_shared (const _Alloc & __a, _Args &&... __args) [inline]`

Create an object that is owned by a `shared_ptr`.

Parameters

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 595 of file `shared_ptr.h`.

2.16.2.2 `template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del* std::get_deleter (const __shared_ptr< _Tp, _Lp > & __p) [inline], [noexcept]`

2.2.3.10 `shared_ptr` `get_deleter` (experimental)

Definition at line 76 of file `shared_ptr.h`.

2.16.2.3 `template<typename _Tp, typename... _Args> shared_ptr<_Tp> std::make_shared (_Args &&... __args) [inline]`

Create an object that is owned by a `shared_ptr`.

Parameters

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<i>std::bad_alloc, or</i>	an exception thrown from the constructor of <code>_Tp</code> .
---------------------------	--

Definition at line 610 of file `shared_ptr.h`.

2.16.2.4 `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>& std::operator<< (std::basic_ostream< _Ch, _Tr > & __os, const __shared_ptr< _Tp, _Lp > & __p) [inline]`

2.2.3.7 `shared_ptr` I/O

Definition at line 66 of file `shared_ptr.h`.

2.17 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct `std::adopt_lock_t`
Assume the calling thread has already obtained mutex ownership and manage it.
- struct `std::defer_lock_t`
Do not acquire ownership of the mutex.
- class `std::lock_guard< _Mutex >`
Scoped lock idiom.
- class `std::mutex`
mutex
- struct `std::once_flag`
once_flag
- class `std::recursive_mutex`
recursive_mutex
- struct `std::try_to_lock_t`
Try to acquire ownership of the mutex without blocking.
- class `std::unique_lock< _Mutex >`
unique_lock

Functions

- `mutex & std::__get_once_mutex ()`
- `void std::__once_proxy (void)`
- `void std::__set_once_func_ptr (unique_lock< mutex > *)`
- `template<typename _Lock >`
`unique_lock< _Lock > std::__try_to_lock (_Lock &__l)`
- `template<typename _Callable, typename... _Args>`
`void std::call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)`
- `template<typename _L1, typename _L2, typename... _L3>`
`void std::lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)`
- `template<typename _Mutex >`
`void std::swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`
`int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`

Variables

- function< void()> **std::__once_functor**
- constexpr adopt_lock_t **std::adopt_lock**
- constexpr defer_lock_t **std::defer_lock**
- constexpr try_to_lock_t **std::try_to_lock**

2.17.1 Detailed Description

Classes for mutex support.

2.17.2 Function Documentation

2.17.2.1 `template<typename _Callable, typename... _Args> void std::call_once (once_flag & __once, _Callable && __f, _Args &&... __args)`

call_once

Definition at line 768 of file mutex.

2.17.2.2 `template<typename _L1, typename _L2, typename... _L3> void std::lock (_L1 & __l1, _L2 & __l2, _L3 &... __l3)`

Generic lock.

Parameters

<code>__l1</code>	Meets Mutex requirements (try_lock() may throw).
<code>__l2</code>	Meets Mutex requirements (try_lock() may throw).
<code>__l3</code>	Meets Mutex requirements (try_lock() may throw).

Exceptions

<i>An</i>	exception thrown by an argument's lock() or try_lock() member.
-----------	--

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to lock(), try_lock() and unlock(). If the call exits via an exception any locks that were obtained will be released.

Definition at line 706 of file mutex.

References std::tie().

2.17.2.3 `template<typename _Mutex > void std::swap (unique_lock< _Mutex > & __x, unique_lock< _Mutex > & __y)`
`[inline], [noexcept]`

Partial specialization for unique_lock objects.

Definition at line 605 of file mutex.

2.17.2.4 `template<typename _Lock1, typename _Lock2, typename... _Lock3> int std::try_lock (_Lock1 & __l1, _Lock2 & __l2, _Lock3 &... __l3)`

Generic try_lock.

Parameters

___/1	Meets Mutex requirements (try_lock() may throw).
___/2	Meets Mutex requirements (try_lock() may throw).
___/3	Meets Mutex requirements (try_lock() may throw).

Returns

Returns -1 if all try_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

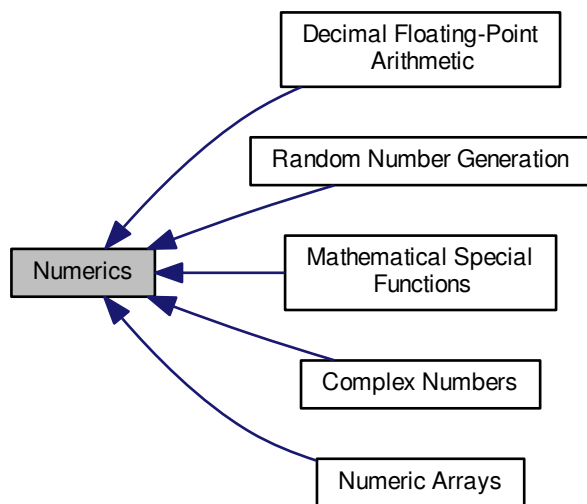
Sequentially calls try_lock() on each argument.

Definition at line 682 of file mutex.

References std::tie().

2.18 Numerics

Collaboration diagram for Numerics:



Modules

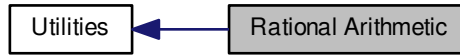
- [Complex Numbers](#)
- [Decimal Floating-Point Arithmetic](#)
- [Mathematical Special Functions](#)
- [Numeric Arrays](#)
- [Random Number Generation](#)

2.18.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

2.19 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Classes

- struct `std::ratio< _Num, _Den >`
Provides compile-time rational arithmetic.
- struct `std::ratio_equal< _R1, _R2 >`
ratio_equal
- struct `std::ratio_not_equal< _R1, _R2 >`
ratio_not_equal

Typedefs

- template<typename _R1, typename _R2 >
using `std::ratio_divide` = typename `__ratio_divide< _R1, _R2 >::type`
- template<typename _R1, typename _R2 >
using `std::ratio_multiply` = typename `__ratio_multiply< _R1, _R2 >::type`
- typedef `ratio< num, den > std::ratio< _Num, _Den >::type`
- typedef `ratio< __safe_multiply
<(_R1::num/__gcd1),(_R2::num/__gcd2)>
::value, __safe_multiply
<(_R1::den/__gcd2),(_R2::den/__gcd1)>
::value > std::__ratio_multiply< _R1, _R2 >::type`
- typedef `__ratio_multiply< _R1,
ratio< _R2::den, _R2::num >
>::type std::__ratio_divide< _R1, _R2 >::type`

Variables

- static constexpr uintmax_t `std::__big_add< __hi1, __lo1, __hi2, __lo2 >::__hi`
- static constexpr uintmax_t `std::__big_sub< __hi1, __lo1, __hi2, __lo2 >::__hi`
- static constexpr uintmax_t `std::__big_mul< __x, __y >::__hi`
- static constexpr uintmax_t `std::__big_add< __hi1, __lo1, __hi2, __lo2 >::__lo`
- static constexpr uintmax_t `std::__big_sub< __hi1, __lo1, __hi2, __lo2 >::__lo`
- static constexpr uintmax_t `std::__big_mul< __x, __y >::__lo`
- static constexpr uintmax_t `std::__big_div_impl< __n1, __n0, __d >::__quot`
- static constexpr uintmax_t `std::__big_div< __n1, __n0, __d >::__quot_hi`
- static constexpr uintmax_t `std::__big_div< __n1, __n0, __d >::__quot_lo`

- static constexpr uintmax_t **std::__big_div_impl**< __n1, __n0, __d >::__rem
- static constexpr uintmax_t **std::__big_div**< __n1, __n0, __d >::__rem
- static constexpr intmax_t **std::ratio**< _Num, _Den >::__den
- static constexpr intmax_t **std::__ratio_multiply**< _R1, _R2 >::__den
- static constexpr intmax_t **std::__ratio_divide**< _R1, _R2 >::__den
- static constexpr intmax_t **std::ratio**< _Num, _Den >::__num
- static constexpr intmax_t **std::__ratio_multiply**< _R1, _R2 >::__num
- static constexpr intmax_t **std::__ratio_divide**< _R1, _R2 >::__num
- static const intmax_t **std::__safe_multiply**< _Pn, _Qn >::__value

2.19.1 Detailed Description

Compile time representation of finite rational numbers.

2.19.2 Typedef Documentation

2.19.2.1 `template<typename _R1, typename _R2> using std::ratio_divide = typedef typename __ratio_divide<_R1, _R2>::type`

ratio_divide

Definition at line 336 of file ratio.

2.19.2.2 `template<typename _R1, typename _R2> using std::ratio_multiply = typedef typename __ratio_multiply<_R1, _R2>::type`

ratio_multiply

Definition at line 313 of file ratio.

2.20 Threads

Collaboration diagram for Threads:



Namespaces

- namespace `std::this_thread`

Classes

- struct `std::hash< thread::id >`
std::hash specialization for thread::id.
- class `std::thread`
thread
- class `std::thread::id`
thread::id

Functions

- `bool std::operator!= (thread::id __x, thread::id __y) noexcept`
- `template<class _CharT, class _Traits > basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)`
- `bool std::operator<= (thread::id __x, thread::id __y) noexcept`
- `bool std::operator> (thread::id __x, thread::id __y) noexcept`
- `bool std::operator>= (thread::id __x, thread::id __y) noexcept`
- `void std::swap (thread &__x, thread &__y) noexcept`

2.20.1 Detailed Description

Classes for thread support.

2.21 Metaprogramming

Collaboration diagram for Metaprogramming:



Classes

- struct `__reflection_typelist< _Elements >`
- struct `common_type< _Tp >`
common_type
- class `result_of< _Signature >`
result_of
- struct `std::add_const< _Tp >`
add_const
- struct `std::add_cv< _Tp >`
add_cv
- struct `std::add_lvalue_reference< _Tp >`
add_lvalue_reference
- struct `std::add_pointer< _Tp >`
add_pointer
- struct `std::add_volatile< _Tp >`
add_volatile
- struct `std::aligned_storage< _Len, _Align >`
Alignment type.
- struct `std::alignment_of< _Tp >`
alignment_of
- class `std::decay< _Tp >`
decay
- struct `std::enable_if< bool, _Tp >`
Define a member typedef type only if a boolean constant is true.
- struct `std::has_trivial_copy_assign< _Tp >`
has_trivial_copy_assign (temporary legacy)
- struct `std::has_trivial_copy_constructor< _Tp >`
has_trivial_copy_constructor (temporary legacy)
- struct `std::has_trivial_default_constructor< _Tp >`
has_trivial_default_constructor (temporary legacy)
- struct `std::has_virtual_destructor< _Tp >`
has_virtual_destructor
- struct `std::integral_constant< _Tp, __v >`
integral_constant
- struct `std::is_abstract< _Tp >`

- is_abstract*
- struct `std::is_arithmetic< _Tp >`
- is_arithmetic*
- struct `std::is_array< typename >`
- is_array*
- struct `std::is_assignable< _Tp, _Up >`
- is_assignable*
- struct `std::is_class< _Tp >`
- is_class*
- struct `std::is_compound< _Tp >`
- is_compound*
- struct `std::is_const< typename >`
- is_const*
- struct `std::is_constructible< _Tp, _Args >`
- is_constructible*
- struct `std::is_convertible< _From, _To >`
- is_convertible*
- struct `std::is_copy_assignable< _Tp >`
- is_copy_assignable*
- struct `std::is_copy_constructible< _Tp >`
- is_copy_constructible*
- struct `std::is_default_constructible< _Tp >`
- is_default_constructible*
- struct `std::is_destructible< _Tp >`
- is_destructible*
- struct `std::is_empty< _Tp >`
- is_empty*
- struct `std::is_enum< _Tp >`
- is_enum*
- struct `std::is_floating_point< _Tp >`
- is_floating_point*
- struct `std::is_fundamental< _Tp >`
- is_fundamental*
- struct `std::is_integral< _Tp >`
- is_integral*
- struct `std::is_literal_type< _Tp >`
- is_literal_type*
- struct `std::is_lvalue_reference< typename >`
- is_lvalue_reference*
- struct `std::is_member_function_pointer< _Tp >`
- is_member_function_pointer*
- struct `std::is_member_object_pointer< _Tp >`
- is_member_object_pointer*
- struct `std::is_move_assignable< _Tp >`
- is_move_assignable*
- struct `std::is_move_constructible< _Tp >`
- is_move_constructible*

- struct `std::is_nothrow_assignable< _Tp, _Up >`
is_nothrow_assignable
- struct `std::is_nothrow_constructible< _Tp, _Args >`
is_nothrow_constructible
- struct `std::is_nothrow_copy_assignable< _Tp >`
is_nothrow_copy_assignable
- struct `std::is_nothrow_copy_constructible< _Tp >`
is_nothrow_copy_constructible
- struct `std::is_nothrow_default_constructible< _Tp >`
is_nothrow_default_constructible
- struct `std::is_nothrow_destructible< _Tp >`
is_nothrow_destructible
- struct `std::is_nothrow_move_assignable< _Tp >`
is_nothrow_move_assignable
- struct `std::is_nothrow_move_constructible< _Tp >`
is_nothrow_move_constructible
- struct `std::is_object< _Tp >`
is_object
- struct `std::is_pod< _Tp >`
is_pod
- struct `std::is_pointer< _Tp >`
is_pointer
- struct `std::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference
- struct `std::is_scalar< _Tp >`
is_scalar
- struct `std::is_signed< _Tp >`
is_signed
- struct `std::is_standard_layout< _Tp >`
is_standard_layout
- struct `std::is_trivial< _Tp >`
is_trivial
- struct `std::is_trivially_destructible< _Tp >`
is_trivially_constructible (still unimplemented)
- struct `std::is_union< _Tp >`
is_union
- struct `std::is_unsigned< _Tp >`
is_unsigned
- struct `std::is_void< _Tp >`
is_void
- struct `std::is_volatile< typename >`
is_volatile
- struct `std::make_signed< _Tp >`

- make_signed*
- struct `std::make_unsigned< _Tp >`
 - make_unsigned*
- struct `std::rank< typename >`
 - rank*
- class `std::reference_wrapper< _Tp >`
 - Primary class template for reference_wrapper.*
- struct `std::remove_const< _Tp >`
 - remove_const*
- struct `std::remove_extent< _Tp >`
 - remove_extent*
- struct `std::remove_pointer< _Tp >`
 - remove_pointer*
- struct `std::remove_volatile< _Tp >`
 - remove_volatile*
- struct `std::tr2::__reflection_typelist< _First, _Rest...>`
 - Partial specialization.*
- struct `std::tr2::__reflection_typelist<>`
 - Specialization for an empty typelist.*
- struct `std::tr2::bases< _Tp >`
 - Sequence abstraction metafunctions for manipulating a typelist.*
- struct `std::tr2::direct_bases< _Tp >`
 - Enumerate all the direct base classes of a class. Form of a typelist.*
- struct `std::underlying_type< _Tp >`
 - The underlying type of an enum.*

Typedefs

- `template<typename... _Cond>`
`using std::Require = typename enable_if< __and< _Cond...>::value >::type`
- `typedef integral_constant`
`< bool, false > std::false_type`
- `typedef integral_constant`
`< bool, true > std::true_type`

Variables

- `static constexpr _Tp std::integral_constant< _Tp, __v >::value`

2.21.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

2.21.2 Typedef Documentation

2.21.2.1 `typedef integral_constant<bool, false> std::false_type`

The type used as a compile-time boolean with false value.

Definition at line 72 of file `type_traits`.

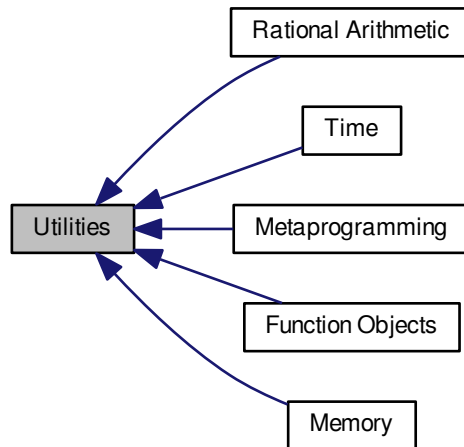
2.21.2.2 `typedef integral_constant<bool, true> std::true_type`

The type used as a compile-time boolean with true value.

Definition at line 69 of file `type_traits`.

2.22 Utilities

Collaboration diagram for Utilities:



Modules

- [Function Objects](#)
- [Memory](#)
- [Metaprogramming](#)
- [Rational Arithmetic](#)
- [Time](#)

Classes

- `struct _Tuple_impl< _Idx, _Elements >`
- `struct std::_Tuple_impl< _Idx >`
- `struct std::_Tuple_impl< _Idx, _Head, _Tail...>`
- `struct std::pair< _T1, _T2 >`
Struct holding two objects of arbitrary type.
- `struct std::piecewise_construct_t`
piecewise_construct_t
- `class std::tuple< _Elements >`
Primary class template, tuple.
- `class std::tuple< _T1, _T2 >`
Partial specialization, 2-element tuple. Includes construction and assignment from a pair.
- `struct std::tuple_element< 0, tuple< _Head, _Tail...> >`
- `struct std::tuple_element< __i, tuple< _Head, _Tail...> >`
- `struct std::tuple_size< tuple< _Elements...> >`

- class `tuple_size`
- struct `std::type_index`

Class `type_index`
The class `type_index` provides a simple wrapper for `type_info` which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).
- struct `std::uses_allocator< tuple< _Types...>, _Alloc >`

Partial specialization for tuples.
- struct `tuple_element< __i, _Tp >`

Gives the type of the *i*th element of a given tuple type.
- struct `tuple_size< _Tp >`

Finds the size of a given tuple type.

Typedefs

- template<typename `_Tp` >
 using `std::__empty_not_final` = typename conditional< __is_final(`_Tp`), false_type, __is_empty_non_tuple< `_Tp` >>::type

Functions

- template<typename... `_Args1`, typename... `_Args2`>
`std::pair< _T1, _T2 >::pair` (`piecewise_construct_t`, `tuple< _Args1...>`, `tuple< _Args2...>`)
- template<typename `_Tp` >
`_Tp * std::__addressof` (`_Tp &__r`) noexcept
- template<std::size_t `__i`, typename `_Head`, typename... `_Tail`>
 constexpr `__add_ref< _Head >::type std::__get_helper` (`_Tuple_impl< __i, _Head, _Tail...> &__t`) noexcept
- template<std::size_t `__i`, typename `_Head`, typename... `_Tail`>
 constexpr `__add_c_ref< _Head >::type std::__get_helper` (`const _Tuple_impl< __i, _Head, _Tail...> &__t`) noexcept
- template<typename `_Tp` >
`_Tp * std::addressof` (`_Tp &__r`) noexcept
- template<typename `_Tp` >
`add_rvalue_reference< _Tp >::type std::declval` () noexcept
- template<typename `_Tp` >
 constexpr `_Tp && std::forward` (typename `std::remove_reference< _Tp >::type &__t`) noexcept
- template<typename `_Tp` >
 constexpr `_Tp && std::forward` (typename `std::remove_reference< _Tp >::type &&__t`) noexcept
- template<typename... `_Elements`>
`tuple< _Elements &&...> std::forward_as_tuple` (`_Elements &&...__args`) noexcept
- template<std::size_t `__i`, typename... `_Elements`>
 constexpr `__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type std::get` (`tuple< _Elements...> &__t`) noexcept
- template<std::size_t `__i`, typename... `_Elements`>
 constexpr `__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type std::get` (`const tuple< _Elements...> &__t`) noexcept

- `template<std::size_t __i, typename... _Elements>`
`constexpr __add_r_ref`
`< typename tuple_element< __i,`
`tuple< _Elements...> >::type >`
`::type std::get (tuple< _Elements...> &&__t) noexcept`
- `template<class _T1, class _T2 >`
`constexpr pair< typename`
`__decay_and_strip< _T1 >`
`::__type, typename`
`__decay_and_strip< _T2 >`
`::__type > std::make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename... _Elements>`
`constexpr tuple< typename`
`__decay_and_strip< _Elements >`
`::__type...> std::make_tuple (_Elements &&...__args)`
- `template<typename _Tp >`
`constexpr`
`std::remove_reference< _Tp >`
`::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp >`
`constexpr conditional`
`< __move_if_noexcept_cond< _Tp >`
`::value, const _Tp &, _Tp && >`
`::type std::move_if_noexcept (_Tp &__x) noexcept`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator!= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator== (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator>= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _T1, class _T2 >`
`void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >`

- `template<typename _Tp, size_t _Nm>`
`void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a`
- `template<typename... _Elements>`
`void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`
`tuple< _Elements &...> std::tie (_Elements &... __args) noexcept`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`
`constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::__type`

Variables

- **`std::__a`**
- `void * std::__b`
- `void`
`is_nothrow_move_assignable`
`< _Tp >::value _Tp std::__tmp`
- `const _Swallow_assign std::ignore`
- `constexpr piecewise_construct_t std::piecewise_construct`

2.22.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

2.22.2 Function Documentation

2.22.2.1 `template<typename _Tp> _Tp* std::__addressof (_Tp & __r) [inline], [noexcept]`

Same as C++11 `std::addressof`.

Definition at line 47 of file `move.h`.

Referenced by `std::addressof()`, `std::begin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > >::data()`, `std::end()`, `std::forward_list< _Tp, _Alloc >::remove()`, and `std::list< _Tp, _Alloc >::remove()`.

2.22.2.2 `template<typename _Tp> _Tp* std::addressof (_Tp & __r) [inline], [noexcept]`

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

Parameters

<code>__r</code>	Reference to an object or function.
------------------	-------------------------------------

Returns

The actual address.

Definition at line 135 of file `move.h`.

References `std::__addressof()`.

Referenced by `std::pointer_traits< _Tp * >::pointer_to()`.

2.22.2.3 `template<typename _Tp> add_rvalue_reference<_Tp>::type std::declval () [inline], [noexcept]`

Utility to simplify expressions used in unevaluated operands.

Definition at line 1869 of file `type_traits`.

2.22.2.4 `template<typename _Tp> constexpr _Tp&& std::forward (typename std::remove_reference<_Tp>::type & _t) [noexcept]`

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 76 of file `move.h`.

2.22.2.5 `template<typename _Tp> constexpr _Tp&& std::forward (typename std::remove_reference<_Tp>::type && _t) [noexcept]`

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 87 of file `move.h`.

2.22.2.6 `template<class _T1, class _T2> constexpr pair<typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type> std::make_pair (_T1 && __x, _T2 && __y)`

A convenience wrapper for creating a pair from two objects.

Parameters

<code>__x</code>	The first object.
<code>__y</code>	The second object.

Returns

A newly-constructed `pair<>` object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 276 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_debug::__get_distance()`, `__gnu_parallel::__parallel_merge_advance()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__find_if_selector::__M_sequential_algorithm()`, `__gnu_parallel::__adjacent_find_selector::__M_sequential_algorithm()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm()`, `std::minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms_pu()`, and `__gnu_pbds::detail::pat_trie_base::__Node_citer<Node, Leaf, Head, Inode, _C_iterator, Iterator, _Alloc>::valid_prefix()`.

2.22.2.7 `template<typename _Tp > constexpr std::remove_reference<_Tp>::type&& std::move (_Tp && __t)`
`[noexcept]`

Convert a value to an rvalue.

Parameters

<code>__t</code>	A thing of arbitrary type.
------------------	----------------------------

Returns

The parameter cast to an rvalue-reference to allow moving it.

Definition at line 101 of file `move.h`.

2.22.2.8 `template<typename _Tp > constexpr conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp&, _Tp&&>::type`
`std::move_if_noexcept (_Tp & __x)` `[inline]`, `[noexcept]`

Conditionally convert a value to an rvalue.

Parameters

<code>__x</code>	A thing of arbitrary type.
------------------	----------------------------

Returns

The parameter, possibly cast to an rvalue-reference.

Same as `std::move` unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Definition at line 121 of file `move.h`.

2.22.2.9 `template<class _T1, class _T2 > constexpr bool std::operator!= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)` `[inline]`

Uses `operator==` to find the result.

Definition at line 227 of file `stl_pair.h`.

2.22.2.10 `template<class _T1, class _T2 > constexpr bool std::operator< (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)` `[inline]`

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

Definition at line 220 of file `stl_pair.h`.

References `std::pair< _T1, _T2 >::first`.

2.22.2.11 `template<class _T1, class _T2 > constexpr bool std::operator<= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)` `[inline]`

Uses `operator<` to find the result.

Definition at line 239 of file `stl_pair.h`.

2.22.2.12 `template<class _T1, class _T2> constexpr bool std::operator==(const pair< _T1, _T2> & __x, const pair< _T1, _T2> & __y) [inline]`

Two pairs of the same type are equal iff their members are equal.

Definition at line 214 of file `stl_pair.h`.

References `std::pair< _T1, _T2>::first`, and `std::pair< _T1, _T2>::second`.

2.22.2.13 `template<class _T1, class _T2> constexpr bool std::operator> (const pair< _T1, _T2> & __x, const pair< _T1, _T2> & __y) [inline]`

Uses `operator<` to find the result.

Definition at line 233 of file `stl_pair.h`.

2.22.2.14 `template<class _T1, class _T2> constexpr bool std::operator>= (const pair< _T1, _T2> & __x, const pair< _T1, _T2> & __y) [inline]`

Uses `operator<` to find the result.

Definition at line 245 of file `stl_pair.h`.

2.22.2.15 `template<class _T1, class _T2> void std::swap (pair< _T1, _T2> & __x, pair< _T1, _T2> & __y) [inline], [noexcept]`

See `std::pair::swap()`.

Definition at line 254 of file `stl_pair.h`.

2.22.2.16 `template<typename _Tp> void std::swap (_Tp & __a, _Tp & __b) const [inline], [noexcept]`

Swaps two values.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

Nothing.

2.22.2.17 `template<typename _Tp, size_t _Nm> void std::swap (_Tp(&) __a[_Nm], _Tp(&) __b[_Nm]) [inline], [noexcept]`

Swap the contents of two arrays.

2.22.2.18 `template<typename... _Elements> void std::swap (tuple< _Elements...> & __x, tuple< _Elements...> & __y) [inline], [noexcept]`

`swap`

Definition at line 1048 of file `tuple`.

2.22.2.19 `template<typename... _Elements> tuple<_Elements&...> std::tie (_Elements &... __args) [inline], [noexcept]`

`tie`

Definition at line 1042 of file tuple.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::lock()`, and `std::try_lock()`.

2.22.2.20 `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>
constexpr auto std::tuple_cat (_Tpls &&... __tps) -> typename __tuple_cat_result<_Tpls...>::__type`

`tuple_cat`

Definition at line 1030 of file tuple.

2.22.3 Variable Documentation

2.22.3.1 `constexpr piecewise_construct_t std::piecewise_construct`

`piecewise_construct`

Definition at line 79 of file `stl_pair.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc>::operator[]()`.

2.23 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class `std::gslice`
Class defining multi-dimensional subset of an array.
- class `std::gslice_array< _Tp >`
Reference to multi-dimensional subset of an array.
- class `std::indirect_array< _Tp >`
Reference to arbitrary subset of an array.
- class `std::mask_array< _Tp >`
Reference to selected subset of an array.
- class `std::slice`
Class defining one-dimensional subset of an array.
- class `std::slice_array< _Tp >`
Reference to one-dimensional subset of an array.
- class `std::valarray< _Tp >`
Smart array designed to support numeric processing.

Macros

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array< _Tp >::gslice_array (const gslice_array &)`
- `std::indirect_array< _Tp >::indirect_array (const indirect_array &)`

- `std::mask_array<_Tp>::mask_array` (const mask_array &)
- `std::slice::slice` ()
- `std::slice::slice` (size_t __o, size_t __d, size_t __s)
- `std::slice_array<_Tp>::slice_array` (const slice_array &)
- `std::valarray<_Tp>::valarray` ()
- `std::valarray<_Tp>::valarray` (size_t)
- `std::valarray<_Tp>::valarray` (const _Tp &, size_t)
- `std::valarray<_Tp>::valarray` (const valarray &)
- `std::valarray<_Tp>::valarray` (valarray &&) noexcept
- `std::valarray<_Tp>::valarray` (const slice_array<_Tp> &)
- `std::valarray<_Tp>::valarray` (const gslice_array<_Tp> &)
- `std::valarray<_Tp>::valarray` (const mask_array<_Tp> &)
- `std::valarray<_Tp>::valarray` (const indirect_array<_Tp> &)
- `std::valarray<_Tp>::valarray` (initializer_list<_Tp>)
- template<class _Dom>
 - `std::valarray<_Tp>::valarray`** (const _Expr<_Dom, _Tp> &__e)
- template<typename _Tp>
 - `std::valarray<_Tp>::valarray`** (const _Tp *__restrict __p, size_t __n)
- `std::gslice::~gslice` ()
- _Expr<_ValFunClos<_ValArray, _Tp>, _Tp> `std::valarray<_Tp>::apply` (_Tp func(_Tp)) const
- _Expr<_RefFunClos<_ValArray, _Tp>, _Tp> `std::valarray<_Tp>::apply` (_Tp func(const _Tp &)) const
- template<class _Tp>
 - `_Tp * std::begin` (valarray<_Tp> &__va)
- template<class _Tp>
 - const _Tp * `std::begin` (const valarray<_Tp> &__va)
- `valarray<_Tp> std::valarray<_Tp>::cshift` (int __n) const
- template<class _Tp>
 - `_Tp * std::end` (valarray<_Tp> &__va)
- template<class _Tp>
 - const _Tp * `std::end` (const valarray<_Tp> &__va)
- `_Tp std::valarray<_Tp>::max` () const
- `_Tp std::valarray<_Tp>::min` () const
- `_UnaryOp<__logical_not>::_Rt std::valarray<_Tp>::operator!` () const
- template<typename _Tp>
 - _Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type> **`std::operator!=`** (const valarray<_Tp> &__v, const _Tp &__t)
- template<typename _Tp>
 - _Expr<_BinClos<__not_equal_to, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type> **`std::operator!=`** (const _Tp &__t, const valarray<_Tp> &__v)

- `template<typename _Tp >`
`_Expr< _BinClos`
`< __not_equal_to, _ValArray,`
`_ValArray, _Tp, _Tp >`
`, typename __fun`
`< __not_equal_to, _Tp >`
`::result_type > std::operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __modulus,`
`_Tp >::result_type > std::operator% (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __modulus,`
`_Tp >::result_type > std::operator% (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __modulus,`
`_Tp >::result_type > std::operator% (const _Tp &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator%= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator%= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator%= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_and, _Tp >`
`::result_type > std::operator& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_and, _Tp >`
`::result_type > std::operator& (const _Tp &__t, const valarray< _Tp > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_and, _Tp >`
`::result_type > std::operator& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __logical_and, _Tp >`
`::result_type > std::operator&& (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun`
`< __logical_and, _Tp >`
`::result_type > std::operator&& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __logical_and, _Tp >`
`::result_type > std::operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator&= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator&= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator&= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator&= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator&= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator&= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator&= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __multiplies,`
`_Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __multiplies,`
`_Tp >::result_type > std::operator* (const _Tp &__t, const valarray< _Tp > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __multiplies,`
`_Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator*= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator*= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator*= (const _Expr< _Dom, _Tp > &)`
- `_UnaryOp< __unary_plus >::Rt std::valarray< _Tp >::operator+ () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __plus, _Tp >`
`::result_type > std::operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __plus, _Tp >`
`::result_type > std::operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __plus, _Tp >`
`::result_type > std::operator+ (const _Tp &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator+= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator+= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator+= (const _Expr< _Dom, _Tp > &)`

- `_UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- () const`
- `template<typename _Tp >
_Expr< _BinClos< __minus,
_ValArray, _ValArray, _Tp, _Tp >
, typename __fun< __minus, _Tp >
::result_type > std::operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >
_Expr< _BinClos< __minus,
_ValArray, _Constant, _Tp, _Tp >
, typename __fun< __minus, _Tp >
::result_type > std::operator- (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >
_Expr< _BinClos< __minus,
_Constant, _ValArray, _Tp, _Tp >
, typename __fun< __minus, _Tp >
::result_type > std::operator- (const _Tp &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `template<class _Dom >
void std::gslice_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >
void std::indirect_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >
void std::mask_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `template<class _Dom >
void std::slice_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator= (const valarray< _Tp > &)`
- `template<class _Dom >
valarray< _Tp > & std::valarray< _Tp >::operator= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >
_Expr< _BinClos< __divides,
_ValArray, _ValArray, _Tp, _Tp >
, typename __fun< __divides,
_Tp >::result_type > std::operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >
_Expr< _BinClos< __divides,
_ValArray, _Constant, _Tp, _Tp >
, typename __fun< __divides,
_Tp >::result_type > std::operator/ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >
_Expr< _BinClos< __divides,
_Constant, _ValArray, _Tp, _Tp >
, typename __fun< __divides,
_Tp >::result_type > std::operator/ (const _Tp &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator/= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator/= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator/= (const valarray< _Tp > &) const`
- `template<class _Dom >
void std::gslice_array< _Tp >::operator/= (const _Expr< _Dom, _Tp > &) const`

- `template<class _Dom >`
`void std::mask_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator/= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator/= (const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator/= (const valarray<_Tp> &)`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator/= (const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun<__less, _Tp >`
`::result_type > std::operator< (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun<__less, _Tp >`
`::result_type > std::operator< (const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun<__less, _Tp >`
`::result_type > std::operator< (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__shift_left,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun<__shift_left,`
`_Tp>::result_type > std::operator<< (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__shift_left,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun<__shift_left,`
`_Tp>::result_type > std::operator<< (const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__shift_left,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun<__shift_left,`
`_Tp>::result_type > std::operator<< (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `void std::gslice_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`

- `valarray<_Tp> & std::valarray<_Tp>::operator<=<= (const _Tp &`
- `valarray<_Tp> & std::valarray<_Tp>::operator<=<= (const valarray<_Tp> &`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator<=<= (const _Expr<_Dom, _Tp> &`
- `template<typename _Tp >`
`_Expr<_BinClos<__less_equal,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun<__less_equal,`
`_Tp>::result_type > std::operator<=<= (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less_equal,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun<__less_equal,`
`_Tp>::result_type > std::operator<=<= (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less_equal,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun<__less_equal,`
`_Tp>::result_type > std::operator<=<= (const valarray<_Tp> &__v, const _Tp &__t)`
- `gslice_array & std::gslice_array<_Tp>::operator= (const gslice_array &`
- `indirect_array & std::indirect_array<_Tp>::operator= (const indirect_array &`
- `mask_array & std::mask_array<_Tp>::operator= (const mask_array &`
- `void std::gslice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `gslice & std::gslice::operator= (const gslice &`
- `void std::gslice_array<_Tp>::operator= (const _Tp &) const`
- `void std::mask_array<_Tp>::operator= (const _Tp &) const`
- `void std::indirect_array<_Tp>::operator= (const _Tp &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `slice_array & std::slice_array<_Tp>::operator= (const slice_array &`
- `void std::slice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::slice_array<_Tp>::operator= (const _Tp &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Ex >`
`void std::mask_array<_Tp>::operator= (const _Expr<_Ex, _Tp> &__e) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const valarray<_Tp> &__v)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (valarray<_Tp> &&__v) noexcept`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const _Tp &__t)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const slice_array<_Tp> &__sa)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const gslice_array<_Tp> &__ga)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const mask_array<_Tp> &__ma)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const indirect_array<_Tp> &__ia)`
- `valarray & std::valarray<_Tp>::operator= (initializer_list<_Tp> __l)`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator= (const _Expr<_Dom, _Tp> &`

- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __equal_to,`
`_Tp >::result_type > std::operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __equal_to,`
`_Tp >::result_type > std::operator== (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __equal_to,`
`_Tp >::result_type > std::operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __greater,`
`_Tp >::result_type > std::operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __greater,`
`_Tp >::result_type > std::operator> (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __greater,`
`_Tp >::result_type > std::operator> (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos`
`< __greater_equal, _Constant,`
`_ValArray, _Tp, _Tp >`
`, typename __fun`
`< __greater_equal, _Tp >`
`::result_type > std::operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos`
`< __greater_equal, _ValArray,`
`_Constant, _Tp, _Tp >`
`, typename __fun`
`< __greater_equal, _Tp >`
`::result_type > std::operator>= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos`
`< __greater_equal, _ValArray,`
`_ValArray, _Tp, _Tp >`
`, typename __fun`
`< __greater_equal, _Tp >`
`::result_type > std::operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`

```

    _Expr< _BinClos< __shift_right,
    _ValArray, _Constant, _Tp, _Tp >
    , typename __fun
    < __shift_right, _Tp >
    ::result_type > std::operator>> (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
    _Expr< _BinClos< __shift_right,
    _Constant, _ValArray, _Tp, _Tp >
    , typename __fun
    < __shift_right, _Tp >
    ::result_type > std::operator>> (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
    _Expr< _BinClos< __shift_right,
    _ValArray, _ValArray, _Tp, _Tp >
    , typename __fun
    < __shift_right, _Tp >
    ::result_type > std::operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• void std::gslice_array< _Tp >::operator>>= (const valarray< _Tp > &) const
• void std::mask_array< _Tp >::operator>>= (const valarray< _Tp > &) const
• void std::indirect_array< _Tp >::operator>>= (const valarray< _Tp > &) const
• template<class _Dom >
    void std::gslice_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const
• template<class _Dom >
    void std::indirect_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const
• template<class _Dom >
    void std::mask_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const
• void std::slice_array< _Tp >::operator>>= (const valarray< _Tp > &) const
• template<class _Dom >
    void std::slice_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const
• valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Tp &)
• valarray< _Tp > & std::valarray< _Tp >::operator>>= (const valarray< _Tp > &)
• template<class _Dom >
    valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &)
• _Tp & std::valarray< _Tp >::operator[] (size_t __i)
• const _Tp & std::valarray< _Tp >::operator[] (size_t) const
• _Expr< _SClos< _ValArray, _Tp >
    , _Tp > std::valarray< _Tp >::operator[] (slice __s) const
• slice_array< _Tp > std::valarray< _Tp >::operator[] (slice __s)
• _Expr< _GClos< _ValArray, _Tp >
    , _Tp > std::valarray< _Tp >::operator[] (const gslice &__s) const
• gslice_array< _Tp > std::valarray< _Tp >::operator[] (const gslice &__s)
• valarray< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m) const
• mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m)
• _Expr< _IClos< _ValArray, _Tp >
    , _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i) const
• indirect_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i)
• template<typename _Tp >
    _Expr< _BinClos< __bitwise_xor,
    _ValArray, _ValArray, _Tp, _Tp >
    , typename __fun
    < __bitwise_xor, _Tp >
    ::result_type > std::operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

```

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_xor, _Tp >`
`::result_type > std::operator^ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_xor, _Tp >`
`::result_type > std::operator^ (const _Tp &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator^= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > std::operator| (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator|= (const valarray< _Tp > &) const`

- `template<class _Dom >`
`void std::slice_array<_Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `valarray<_Tp > & std::valarray<_Tp >::operator|= (const _Tp &)`
- `valarray<_Tp > & std::valarray<_Tp >::operator|= (const valarray<_Tp > &)`
- `template<class _Dom >`
`valarray<_Tp > & std::valarray<_Tp >::operator|= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr<_BinClos<__logical_or,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun<__logical_or,`
`_Tp >::result_type > std::operator|| (const valarray<_Tp > &__v, const valarray<_Tp > &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__logical_or,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun<__logical_or,`
`_Tp >::result_type > std::operator|| (const valarray<_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__logical_or,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun<__logical_or,`
`_Tp >::result_type > std::operator|| (const _Tp &__t, const valarray<_Tp > &__v)`
- `_UnaryOp<__bitwise_not >::Rt std::valarray<_Tp >::operator~ () const`
- `void std::valarray<_Tp >::resize (size_t __size, _Tp __c=_Tp())`
- `valarray<_Tp > std::valarray<_Tp >::shift (int __n) const`
- `size_t std::slice::size () const`
- `valarray<size_t > std::gslice::size () const`
- `size_t std::valarray<_Tp >::size () const`
- `size_t std::slice::start () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::stride () const`
- `valarray<size_t > std::gslice::stride () const`
- `_Tp std::valarray<_Tp >::sum () const`
- `void std::valarray<_Tp >::swap (valarray<_Tp > &__v) noexcept`

2.23.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

2.23.2 Function Documentation

2.23.2.1 `std::gslice::gslice ()` `[inline]`

Construct an empty slice.

Definition at line 149 of file `gslice.h`.

2.23.2.2 `std::gslice::gslice (size_t __o, const valarray<size_t> &__l, const valarray<size_t> &__s)` `[inline]`

Construct a slice.

Constructs a slice with as many dimensions as the length of the `l` and `s` arrays.

Parameters

<code>__o</code>	Offset in array of first element.
<code>__l</code>	Array of dimension lengths.
<code>__s</code>	Array of dimension strides between array elements.

Definition at line 153 of file `gslice.h`.

2.23.2.3 `std::gslice::gslice (const gslice & __g) [inline]`

Copy constructor.

Definition at line 158 of file `gslice.h`.

2.23.2.4 `template<typename _Tp> gslice_array<_Tp>::gslice_array (const gslice_array<_Tp> & __a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file `gslice_array.h`.

2.23.2.5 `template<typename _Tp> indirect_array<_Tp>::indirect_array (const indirect_array<_Tp> & __a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file `indirect_array.h`.

2.23.2.6 `template<typename _Tp> mask_array<_Tp>::mask_array (const mask_array<_Tp> & a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 139 of file `mask_array.h`.

2.23.2.7 `std::slice::slice () [inline]`

Construct an empty slice.

Definition at line 90 of file `slice_array.h`.

2.23.2.8 `std::slice::slice (size_t __o, size_t __d, size_t __s) [inline]`

Construct a slice.

Parameters

<code>__o</code>	Offset in array of first element.
<code>__d</code>	Number of elements in slice.
<code>__s</code>	Stride between array elements.

Definition at line 94 of file `slice_array.h`.

2.23.2.9 `template<typename _Tp> slice_array<_Tp>::slice_array (const slice_array<_Tp> & a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 207 of file `slice_array.h`.

2.23.2.10 `template<typename _Tp> valarray<_Tp>::valarray () [inline]`

Construct an empty array.

Definition at line 605 of file `valarray`.

2.23.2.11 `template<typename _Tp> valarray<_Tp>::valarray (size_t __n) [inline],[explicit]`

Construct an array with n elements.

Definition at line 609 of file `valarray`.

2.23.2.12 `template<typename _Tp> valarray<_Tp>::valarray (const _Tp & __t, size_t __n) [inline]`

Construct an array with n elements initialized to t .

Definition at line 615 of file `valarray`.

2.23.2.13 `template<typename _Tp> valarray<_Tp>::valarray (const valarray<_Tp> & __v) [inline]`

Copy constructor.

Definition at line 630 of file `valarray`.

2.23.2.14 `template<typename _Tp> valarray<_Tp>::valarray (valarray<_Tp> && __v) [inline],[noexcept]`

Move constructor.

Definition at line 638 of file `valarray`.

2.23.2.15 `template<typename _Tp> valarray<_Tp>::valarray (const slice_array<_Tp> & __sa) [inline]`

Construct an array with the same size and values in sa .

Definition at line 648 of file `valarray`.

2.23.2.16 `template<typename _Tp> valarray<_Tp>::valarray (const gslice_array<_Tp> & __ga) [inline]`

Construct an array with the same size and values in ga .

Definition at line 657 of file `valarray`.

2.23.2.17 `template<typename _Tp> valarray<_Tp>::valarray (const mask_array<_Tp> & __ma) [inline]`

Construct an array with the same size and values in ma .

Definition at line 668 of file `valarray`.

2.23.2.18 `template<typename _Tp> valarray<_Tp>::valarray (const indirect_array<_Tp> & __ia) [inline]`

Construct an array with the same size and values in ia .

Definition at line 677 of file `valarray`.

2.23.2.19 `template<typename _Tp> valarray<_Tp>::valarray (initializer_list<_Tp> __l) [inline]`

Construct an array with an `initializer_list` of values.

Definition at line 687 of file `valarray`.

2.23.2.20 `std::gslice::~gslice () [inline]`

Destructor.

Definition at line 163 of file `gslice.h`.

2.23.2.21 `template<class _Tp> _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::apply (_Tp func_Tp) const [inline]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

<i>func</i>	Function of Tp returning Tp to apply.
-------------	---------------------------------------

Returns

New valarray with transformed elements.

Definition at line 1036 of file valarray.

2.23.2.22 `template<class _Tp> _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::apply (_Tp funcconst_Tp &) const [inline]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

<i>func</i>	Function of const Tp& returning Tp to apply.
-------------	--

Returns

New valarray with transformed elements.

Definition at line 1044 of file valarray.

2.23.2.23 `template<class _Tp> _Tp* std::begin (valarray< _Tp > & __va) [inline]`

Return an iterator pointing to the first element of the valarray.

Parameters

<i>__va</i>	valarray.
-------------	-----------

Definition at line 1183 of file valarray.

References std::__addressof().

2.23.2.24 `template<class _Tp> const _Tp* std::begin (const valarray< _Tp > & __va) [inline]`

Return an iterator pointing to the first element of the const valarray.

Parameters

<i>__va</i>	valarray.
-------------	-----------

Definition at line 1193 of file valarray.

References `std::__addressof()`.

2.23.2.25 `template<class _Tp> valarray<_Tp> valarray<_Tp>::cshift(int __n) const [inline]`

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is $(i - n) \% \text{size}()$. The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

<code>__n</code>	Number of element positions to rotate.
------------------	--

Returns

New valarray with elements in shifted positions.

Definition at line 962 of file `valarray`.

2.23.2.26 `template<class _Tp> _Tp* std::end(valarray<_Tp> &__va) [inline]`

Return an iterator pointing to one past the last element of the valarray.

Parameters

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1203 of file `valarray`.

References `std::__addressof()`, and `std::valarray<_Tp>::size()`.

2.23.2.27 `template<class _Tp> const _Tp* std::end(const valarray<_Tp> &__va) [inline]`

Return an iterator pointing to one past the last element of the const valarray.

Parameters

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1213 of file `valarray`.

References `std::__addressof()`, and `std::valarray<_Tp>::size()`.

2.23.2.28 `template<typename _Tp> _Tp valarray<_Tp>::max() const [inline]`

Return the maximum element using operator<().

Definition at line 1028 of file `valarray`.

References `std::max_element()`.

2.23.2.29 `template<typename _Tp> _Tp valarray<_Tp>::min() const [inline]`

Return the minimum element using operator<().

Definition at line 1020 of file `valarray`.

References `std::min_element()`.

2.23.2.30 `template<typename _Tp> valarray< _Tp >::template _UnaryOp< _logical_not >::Rt valarray< _Tp >::operator! ()
const [inline]`

Return a new valarray by applying unary ! to each element.

Definition at line 1063 of file `valarray`.

2.23.2.31 `template<typename _Tp> void gslice_array< _Tp >::operator%=(const valarray< _Tp > & __v) const
[inline]`

Modulo slice elements by corresponding elements of `v`.

Definition at line 202 of file `gslice_array.h`.

2.23.2.32 `template<typename _Tp> void mask_array< _Tp >::operator%=(const valarray< _Tp > & __v) const [inline]`

Modulo slice elements by corresponding elements of `v`.

Definition at line 192 of file `mask_array.h`.

2.23.2.33 `template<typename _Tp> void indirect_array< _Tp >::operator%=(const valarray< _Tp > & __v) const
[inline]`

Modulo slice elements by corresponding elements of `v`.

Definition at line 196 of file `indirect_array.h`.

2.23.2.34 `template<typename _Tp> void slice_array< _Tp >::operator%=(const valarray< _Tp > & __v) const [inline]`

Modulo slice elements by corresponding elements of `v`.

Definition at line 258 of file `slice_array.h`.

2.23.2.35 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator%=(const _Tp & __t) [inline]`

Set each element `e` of array to `e % t`.

Definition at line 1090 of file `valarray`.

2.23.2.36 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator%=(const valarray< _Tp > & __v) [inline]`

Modulo elements of array by corresponding elements of `v`.

Definition at line 1090 of file `valarray`.

2.23.2.37 `template<typename _Tp> void gslice_array< _Tp >::operator&=(const valarray< _Tp > & __v) const [inline]`

Logical and slice elements with corresponding elements of `v`.

Definition at line 206 of file `gslice_array.h`.

2.23.2.38 `template<typename _Tp> void mask_array< _Tp >::operator&=(const valarray< _Tp > & __v) const [inline]`

Logical and slice elements with corresponding elements of `v`.

Definition at line 196 of file `mask_array.h`.

2.23.2.39 `template<typename _Tp> void indirect_array<_Tp>::operator&= (const valarray<_Tp> & __v) const`
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 200 of file `indirect_array.h`.

2.23.2.40 `template<typename _Tp> void slice_array<_Tp>::operator&= (const valarray<_Tp> & __v) const` `[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 262 of file `slice_array.h`.

2.23.2.41 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator&= (const _Tp & __t)` `[inline]`

Set each element *e* of array to *e* & *t*.

Definition at line 1092 of file `valarray`.

2.23.2.42 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator&= (const valarray<_Tp> & __v)` `[inline]`

Logical and corresponding elements of *v* with elements of array.

Definition at line 1092 of file `valarray`.

2.23.2.43 `template<typename _Tp> void gslice_array<_Tp>::operator*= (const valarray<_Tp> & __v) const` `[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 200 of file `gslice_array.h`.

2.23.2.44 `template<typename _Tp> void mask_array<_Tp>::operator*= (const valarray<_Tp> & __v) const` `[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 190 of file `mask_array.h`.

2.23.2.45 `template<typename _Tp> void indirect_array<_Tp>::operator*= (const valarray<_Tp> & __v) const`
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 194 of file `indirect_array.h`.

2.23.2.46 `template<typename _Tp> void slice_array<_Tp>::operator*= (const valarray<_Tp> & __v) const` `[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 256 of file `slice_array.h`.

2.23.2.47 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator*= (const _Tp & __t)` `[inline]`

Multiply each element of array by *t*.

Definition at line 1088 of file `valarray`.

2.23.2.48 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator*= (const valarray<_Tp> & __v)` `[inline]`

Multiply elements of array by corresponding elements of *v*.

Definition at line 1088 of file `valarray`.

2.23.2.49 `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__unary_plus>::Rt valarray<_Tp>::operator+ () const [inline]`

Return a new valarray by applying unary + to each element.

Definition at line 1060 of file valarray.

2.23.2.50 `template<typename _Tp> void gslice_array<_Tp>::operator+=(const valarray<_Tp> &__v) const [inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 203 of file gslice_array.h.

2.23.2.51 `template<typename _Tp> void mask_array<_Tp>::operator+=(const valarray<_Tp> &__v) const [inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 193 of file mask_array.h.

2.23.2.52 `template<typename _Tp> void indirect_array<_Tp>::operator+=(const valarray<_Tp> &__v) const [inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 197 of file indirect_array.h.

2.23.2.53 `template<typename _Tp> void slice_array<_Tp>::operator+=(const valarray<_Tp> &__v) const [inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 259 of file slice_array.h.

2.23.2.54 `template<class _Tp> valarray<_Tp> &valarray<_Tp>::operator+=(const _Tp &__t) [inline]`

Add *t* to each element of array.

Definition at line 1086 of file valarray.

2.23.2.55 `template<class _Tp> valarray<_Tp> &valarray<_Tp>::operator+=(const valarray<_Tp> &__v) [inline]`

Add corresponding elements of *v* to elements of array.

Definition at line 1086 of file valarray.

2.23.2.56 `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__negate>::Rt valarray<_Tp>::operator- () const [inline]`

Return a new valarray by applying unary - to each element.

Definition at line 1061 of file valarray.

2.23.2.57 `template<typename _Tp> void gslice_array<_Tp>::operator-= (const valarray<_Tp> &__v) const [inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 204 of file gslice_array.h.

2.23.2.58 `template<typename _Tp> void mask_array<_Tp>::operator-= (const valarray<_Tp> &__v) const [inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 194 of file mask_array.h.

2.23.2.59 `template<typename _Tp> void indirect_array<_Tp>::operator-= (const valarray<_Tp> & __v) const [inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 198 of file `indirect_array.h`.

2.23.2.60 `template<typename _Tp> void slice_array<_Tp>::operator-= (const valarray<_Tp> & __v) const [inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 260 of file `slice_array.h`.

2.23.2.61 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator-= (const _Tp & __t) [inline]`

Subtract *t* to each element of array.

Definition at line 1087 of file `valarray`.

2.23.2.62 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator-= (const valarray<_Tp> & __v) [inline]`

Subtract corresponding elements of *v* from elements of array.

Definition at line 1087 of file `valarray`.

2.23.2.63 `template<typename _Tp> void gslice_array<_Tp>::operator/= (const valarray<_Tp> & __v) const [inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 201 of file `gslice_array.h`.

2.23.2.64 `template<typename _Tp> void mask_array<_Tp>::operator/= (const valarray<_Tp> & __v) const [inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 191 of file `mask_array.h`.

2.23.2.65 `template<typename _Tp> void indirect_array<_Tp>::operator/= (const valarray<_Tp> & __v) const [inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 195 of file `indirect_array.h`.

2.23.2.66 `template<typename _Tp> void slice_array<_Tp>::operator/= (const valarray<_Tp> & __v) const [inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 257 of file `slice_array.h`.

2.23.2.67 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator/= (const _Tp & __t) [inline]`

Divide each element of array by *t*.

Definition at line 1089 of file `valarray`.

2.23.2.68 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator/= (const valarray<_Tp> & __v) [inline]`

Divide elements of array by corresponding elements of *v*.

Definition at line 1089 of file `valarray`.

2.23.2.69 `template<typename _Tp> void gslice_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const`
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 208 of file `gslice_array.h`.

2.23.2.70 `template<typename _Tp> void mask_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const`
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 198 of file `mask_array.h`.

2.23.2.71 `template<typename _Tp> void indirect_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const`
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 202 of file `indirect_array.h`.

2.23.2.72 `template<typename _Tp> void slice_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const`
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 264 of file `slice_array.h`.

2.23.2.73 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator<<= (const _Tp & __t) [inline]`

Left shift each element *e* of array by *t* bits.

Definition at line 1094 of file `valarray`.

2.23.2.74 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator<<= (const valarray< _Tp > & __v)`
`[inline]`

Left shift elements of array by corresponding elements of *v*.

Definition at line 1094 of file `valarray`.

2.23.2.75 `template<typename _Tp> gslice_array< _Tp > & gslice_array< _Tp >::operator= (const gslice_array< _Tp > & __a) [inline]`

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 148 of file `gslice_array.h`.

2.23.2.76 `template<typename _Tp> indirect_array< _Tp > & indirect_array< _Tp >::operator= (const indirect_array< _Tp > & __a) [inline]`

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file `indirect_array.h`.

2.23.2.77 `template<typename _Tp> mask_array< _Tp > & mask_array< _Tp >::operator= (const mask_array< _Tp > & __a)`
`[inline]`

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 149 of file `mask_array.h`.

2.23.2.78 `template<typename _Tp> void gslice_array< _Tp >::operator= (const valarray< _Tp> & __v) const` [inline]

Assign slice elements to corresponding elements of *v*.

Definition at line 166 of file `gslice_array.h`.

References `std::valarray< _Tp >::size()`.

2.23.2.79 `template<typename _Tp> void indirect_array< _Tp >::operator= (const valarray< _Tp> & __v) const`
[inline]

Assign slice elements to corresponding elements of *v*.

Definition at line 168 of file `indirect_array.h`.

2.23.2.80 `gslice & std::gslice::operator= (const gslice & __g)` [inline]

Assignment operator.

Definition at line 170 of file `gslice.h`.

2.23.2.81 `template<typename _Tp> void gslice_array< _Tp >::operator= (const _Tp & __t) const` [inline]

Assign all slice elements to *t*.

Definition at line 158 of file `gslice_array.h`.

2.23.2.82 `template<typename _Tp> void mask_array< _Tp >::operator= (const _Tp & __t) const` [inline]

Assign all slice elements to *t*.

Definition at line 158 of file `mask_array.h`.

2.23.2.83 `template<typename _Tp> void indirect_array< _Tp >::operator= (const _Tp & __t) const` [inline]

Assign all slice elements to *t*.

Definition at line 163 of file `indirect_array.h`.

2.23.2.84 `template<typename _Tp> slice_array< _Tp> & slice_array< _Tp>::operator= (const slice_array< _Tp> & __a)`
[inline]

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 215 of file `slice_array.h`.

2.23.2.85 `template<typename _Tp> void slice_array< _Tp >::operator= (const valarray< _Tp> & __v) const` [inline]

Assign slice elements to corresponding elements of *v*.

Definition at line 229 of file `slice_array.h`.

2.23.2.86 `template<typename _Tp> void slice_array< _Tp >::operator= (const _Tp & __t) const` [inline]

Assign all slice elements to *t*.

Definition at line 224 of file `slice_array.h`.

2.23.2.87 `template<typename _Tp> valarray< _Tp> & valarray< _Tp>::operator= (const valarray< _Tp> & __v)`
[inline]

Assign elements to an array.

Assign elements of array to values in *v*.

Parameters

<code>__v</code>	Valarray to get values from.
------------------	------------------------------

Definition at line 708 of file `valarray`.

```
2.23.2.88  template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator=( valarray< _Tp > && __v )  [inline],
           [noexcept]
```

Move assign elements to an array.

Move assign elements of array to values in *v*.

Parameters

<code>__v</code>	Valarray to get values from.
------------------	------------------------------

Definition at line 732 of file `valarray`.

```
2.23.2.89  template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator=( const _Tp & __t )  [inline]
```

Assign elements to a value.

Assign all elements of array to *t*.

Parameters

<code>__t</code>	Value for elements.
------------------	---------------------

Definition at line 772 of file `valarray`.

```
2.23.2.90  template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator=( const slice_array< _Tp > & __sa )
           [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

<code>__sa</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 780 of file `valarray`.

```
2.23.2.91  template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator=( const gslice_array< _Tp > & __ga )
           [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

<code>__ga</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 790 of file `valarray`.

References `std::valarray<_Tp>::size()`.

2.23.2.92 `template<typename _Tp> valarray<_Tp> & valarray<_Tp>::operator= (const mask_array<_Tp> & __ma)`
`[inline]`

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

<code>__ma</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 800 of file `valarray`.

2.23.2.93 `template<typename _Tp> valarray<_Tp> & valarray<_Tp>::operator= (const indirect_array<_Tp> & __ia)`
`[inline]`

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

<code>__ia</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 810 of file `valarray`.

2.23.2.94 `template<typename _Tp> valarray<_Tp> & valarray<_Tp>::operator= (initializer_list<_Tp> __l)`
`[inline]`

Assign elements to an `initializer_list`.

Assign elements of array to values in `__l`. Results are undefined if `__l` does not have the same size as this array.

Parameters

<code>__l</code>	<code>initializer_list</code> to get values from.
------------------	---

Definition at line 748 of file `valarray`.

2.23.2.95 `template<typename _Tp> void gslice_array<_Tp>::operator>>= (const valarray<_Tp> & __v) const`
`[inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 209 of file `gslice_array.h`.

2.23.2.96 `template<typename _Tp> void mask_array<_Tp>::operator>>= (const valarray<_Tp> & __v) const`
`[inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 199 of file `mask_array.h`.

2.23.2.97 `template<typename _Tp> void indirect_array<_Tp>::operator>>= (const valarray<_Tp> & __v) const`
`[inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 203 of file indirect_array.h.

2.23.2.98 `template<typename _Tp> void slice_array<_Tp>::operator>>= (const valarray<_Tp> & __v) const`
`[inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 265 of file slice_array.h.

2.23.2.99 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator>>= (const _Tp & __t)` `[inline]`

Right shift each element *e* of array by *t* bits.

Definition at line 1095 of file valarray.

2.23.2.100 `template<class _Tp> valarray<_Tp> & valarray<_Tp>::operator>>= (const valarray<_Tp> & __v)`
`[inline]`

Right shift elements of array by corresponding elements of *v*.

Definition at line 1095 of file valarray.

2.23.2.101 `template<typename _Tp> _Tp & valarray<_Tp>::operator[] (size_t __i)` `[inline]`

Return a reference to the *i*'th array element.

Parameters

<code>__i</code>	Index of element to return.
------------------	-----------------------------

Returns

Reference to the *i*'th element.

Definition at line 576 of file valarray.

2.23.2.102 `template<typename _Tp> _Expr<_SClos<_ValArray, _Tp>, _Tp> valarray<_Tp>::operator[] (slice __s) const`
`[inline]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See Also

slice.

Parameters

<code>__s</code>	The source slice.
------------------	-------------------

Returns

New valarray containing elements in `__s`.

Definition at line 829 of file valarray.

2.23.2.103 `template<typename _Tp> slice_array<_Tp> valarray<_Tp>::operator[](slice __s) [inline]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See Also

slice.

Parameters

<code>__s</code>	The source slice.
------------------	-------------------

Returns

New valarray containing elements in `__s`.

Definition at line 837 of file valarray.

2.23.2.104 `template<typename _Tp> _Expr<_GClos<_ValArray, _Tp>, _Tp> valarray<_Tp>::operator[](const gslice & __s) const [inline]`

Return an array subset.

Returns a slice_array referencing the elements of the array indicated by the slice argument.

See Also

gslice.

Parameters

<code>__s</code>	The source slice.
------------------	-------------------

Returns

Slice_array referencing elements indicated by `__s`.

Definition at line 842 of file valarray.

2.23.2.105 `template<typename _Tp> gslice_array<_Tp> valarray<_Tp>::operator[](const gslice & __s) [inline]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

See Also

gslice.

Parameters

<code>__s</code>	The source gslice.
------------------	--------------------

Returns

New valarray containing elements in `__s`.

Definition at line 851 of file valarray.

2.23.2.106 `template<typename _Tp> valarray<_Tp> valarray<_Tp>::operator[] (const valarray<bool> & __m) const`
`[inline]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

<code>__m</code>	The valarray bitmask.
------------------	-----------------------

Returns

New valarray containing elements indicated by `__m`.

Definition at line 859 of file valarray.

References `std::valarray<_Tp>::size()`.

2.23.2.107 `template<typename _Tp> mask_array<_Tp> valarray<_Tp>::operator[] (const valarray<bool> & __m)`
`[inline]`

Return a reference to an array subset.

Returns a new mask_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

<code>__m</code>	The valarray bitmask.
------------------	-----------------------

Returns

New valarray containing elements indicated by `__m`.

Definition at line 871 of file valarray.

References `std::valarray<_Tp>::size()`.

2.23.2.108 `template<typename _Tp> _Expr<_IClos<_ValArray, _Tp>, _Tp> valarray<_Tp>::operator[] (const valarray<size_t> & __i) const` `[inline]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

<code>__i</code>	The valarray element index list.
------------------	----------------------------------

Returns

New valarray containing elements in `__s`.

Definition at line 882 of file `valarray`.

2.23.2.109 `template<typename _Tp > indirect_array< _Tp > valarray< _Tp >::operator[] (const valarray< size_t > & __i)`
`[inline]`

Return a reference to an array subset.

Returns an `indirect_array` referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned `indirect_array` refers to these elements.

Parameters

<code>__i</code>	The valarray element index list.
------------------	----------------------------------

Returns

`Indirect_array` referencing elements in `__i`.

Definition at line 890 of file `valarray`.

References `std::valarray< _Tp >::size()`.

2.23.2.110 `template<typename _Tp > void gslice_array< _Tp >::operator^= (const valarray< _Tp > & __v) const`
`[inline]`

Logical xor slice elements with corresponding elements of `v`.

Definition at line 205 of file `gslice_array.h`.

2.23.2.111 `template<typename _Tp > void mask_array< _Tp >::operator^= (const valarray< _Tp > & __v) const`
`[inline]`

Logical xor slice elements with corresponding elements of `v`.

Definition at line 195 of file `mask_array.h`.

2.23.2.112 `template<typename _Tp > void indirect_array< _Tp >::operator^= (const valarray< _Tp > & __v) const`
`[inline]`

Logical xor slice elements with corresponding elements of `v`.

Definition at line 199 of file `indirect_array.h`.

2.23.2.113 `template<typename _Tp > void slice_array< _Tp >::operator^= (const valarray< _Tp > & __v) const` `[inline]`

Logical xor slice elements with corresponding elements of `v`.

Definition at line 261 of file `slice_array.h`.

2.23.2.114 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator^= (const _Tp & __t) [inline]`

Set each element e of array to $e \wedge t$.

Definition at line 1091 of file `valarray`.

2.23.2.115 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator^= (const valarray< _Tp > & __v) [inline]`

Logical xor corresponding elements of v with elements of array.

Definition at line 1091 of file `valarray`.

2.23.2.116 `template<typename _Tp > void gslice_array< _Tp >::operator|= (const valarray< _Tp > & __v) const [inline]`

Logical or slice elements with corresponding elements of v .

Definition at line 207 of file `gslice_array.h`.

2.23.2.117 `template<typename _Tp > void mask_array< _Tp >::operator|= (const valarray< _Tp > & __v) const [inline]`

Logical or slice elements with corresponding elements of v .

Definition at line 197 of file `mask_array.h`.

2.23.2.118 `template<typename _Tp > void indirect_array< _Tp >::operator|= (const valarray< _Tp > & __v) const [inline]`

Logical or slice elements with corresponding elements of v .

Definition at line 201 of file `indirect_array.h`.

2.23.2.119 `template<typename _Tp > void slice_array< _Tp >::operator|= (const valarray< _Tp > & __v) const [inline]`

Logical or slice elements with corresponding elements of v .

Definition at line 263 of file `slice_array.h`.

2.23.2.120 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator|= (const _Tp & __t) [inline]`

Set each element e of array to $e | t$.

Definition at line 1093 of file `valarray`.

2.23.2.121 `template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator|= (const valarray< _Tp > & __v) [inline]`

Logical or corresponding elements of v with elements of array.

Definition at line 1093 of file `valarray`.

2.23.2.122 `template<typename _Tp > valarray< _Tp >::template _UnaryOp< __bitwise_not >::Rt valarray< _Tp >::operator~ () const [inline]`

Return a new `valarray` by applying unary \sim to each element.

Definition at line 1062 of file `valarray`.

2.23.2.123 `template<class _Tp> void valarray< _Tp >::resize (size_t __size, _Tp __c = _Tp()) [inline]`

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

Parameters

<code>__size</code>	New array size.
<code>__c</code>	New value for all elements.

Definition at line 1003 of file `valarray`.

2.23.2.124 `template<class _Tp> valarray< _Tp> valarray< _Tp>::shift(int __n) const` `[inline]`

Return a shifted array.

A new `valarray` is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new `valarray` has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

Parameters

<code>__n</code>	Number of element positions to shift.
------------------	---------------------------------------

Returns

New `valarray` with elements in shifted positions.

Definition at line 921 of file `valarray`.

2.23.2.125 `size_t std::slice::size() const` `[inline]`

Return size of slice.

Definition at line 102 of file `slice_array.h`.

2.23.2.126 `valarray< size_t> std::gslice::size() const` `[inline]`

Return array of sizes of slice dimensions.

Definition at line 139 of file `gslice.h`.

2.23.2.127 `template<class _Tp> size_t valarray< _Tp>::size() const` `[inline]`

Return the number of elements in array.

Definition at line 908 of file `valarray`.

Referenced by `std::end()`, `std::gslice_array< _Tp>::operator=()`, `std::valarray< _Tp>::operator=()`, and `std::valarray< _Tp>::operator[]()`.

2.23.2.128 `size_t std::slice::start() const` `[inline]`

Return array offset of first slice element.

Definition at line 98 of file `slice_array.h`.

2.23.2.129 `size_t std::gslice::start() const` `[inline]`

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

2.23.2.130 `size_t std::slice::stride () const [inline]`

Return array stride of slice.

Definition at line 106 of file slice_array.h.

2.23.2.131 `valarray< size_t > std::gslice::stride () const [inline]`

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.

2.23.2.132 `template<class _Tp> _Tp valarray< _Tp >::sum () const [inline]`

Return the sum of all elements in the array.

Accumulates the sum of all elements into a `Tp` using `+=`. The order of adding the elements is unspecified.

Definition at line 913 of file valarray.

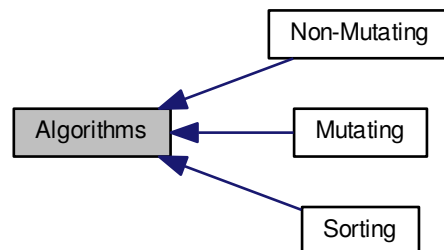
2.23.2.133 `template<class _Tp> void valarray< _Tp >::swap (valarray< _Tp > & __v) [inline], [noexcept]`

Swap.

Definition at line 899 of file valarray.

2.24 Algorithms

Collaboration diagram for Algorithms:



Modules

- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

2.24.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

2.25 Mutating

Collaboration diagram for Mutating:



Functions

- `template<typename _II, typename _OI >`
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate > _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`

2.25.1 Detailed Description

2.25.2 Function Documentation

2.25.2.1 `template<typename _II, typename _OI > _OI std::copy (_II __first, _II __last, _OI __result) [inline]`

Copies the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 450 of file `stl_algobase.h`.

2.25.2.2 `template<typename _BI1, typename _BI2 > _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result) [inline]`

Copies the range [first,last) into result.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

Returns

`result - (first - last)`

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use `copy` instead. Note that the start of the output range may overlap [first,last).

Definition at line 619 of file `stl_algobase.h`.

2.25.2.3 `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::copy_if (`
`_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`

Copy the elements of a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `__result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 974 of file `stl_algo.h`.

2.25.2.4 `template<typename _InputIterator, typename _Size, typename _OutputIterator> _OutputIterator std::copy_n (`
`_InputIterator __first, _Size __n, _OutputIterator __result) [inline]`

Copies the range `[first,first+n)` into `[result,result+n)`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

`result+n`.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1037 of file `stl_algo.h`.

References `std::__iterator_category()`.

2.25.2.5 `template<typename _ForwardIterator, typename _Tp> void std::fill (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp & __value) [inline]`

Fills the range `[first,last)` with copies of `value`.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__value</code>	A reference-to-const of arbitrary type.

Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 721 of file `stl_algobase.h`.

2.25.2.6 `template<typename _OI, typename _Size, typename _Tp > _OI std::fill_n (_OI __first, _Size __n, const _Tp & __value)`
`[inline]`

Fills the range `[first,first+n)` with copies of value.

Parameters

<code>__first</code>	An output iterator.
<code>__n</code>	The count of copies to perform.
<code>__value</code>	A reference-to-const of arbitrary type.

Returns

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 781 of file `stl_algobase.h`.

2.25.2.7 `template<typename _ForwardIterator, typename _Generator > void std::generate (_ForwardIterator __first,`
`_ForwardIterator __last, _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits<_ForwardIterator>::value_type</code>

Returns

`generate()` returns no value.

Performs the assignment `*i = __gen ()` for each `i` in the range `[__first,__last)`.

Definition at line 5071 of file `stl_algo.h`.

2.25.2.8 `template<typename _OutputIterator, typename _Size, typename _Generator > _OutputIterator std::generate_n (`
`_OutputIterator __first, _Size __n, _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__n</code>	The length of the sequence.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits<_ForwardIterator>::value_type</code>

Returns

The end of the sequence, `__first+__n`

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __first+__n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 5102 of file `stl_algo.h`.

2.25.2.9 `template<typename _InputIterator, typename _Predicate> bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Checks whether the sequence is partitioned.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the range `[__first, __last)` is partitioned by `__pred`, i.e. if all elements that satisfy `__pred` appear before those that do not.

Definition at line 826 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

2.25.2.10 `template<typename _ForwardIterator1, typename _ForwardIterator2> void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b) [inline]`

Swaps the contents of two iterators.

Parameters

<code>__a</code>	An iterator.
<code>__b</code>	Another iterator.

Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 119 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::next_permutation()`, `std::prev_permutation()`, `std::random_shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

2.25.2.11 `template<typename _II, typename _OI> _OI std::move (_II __first, _II __last, _OI __result) [inline]`

Moves the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 483 of file `stl_algobase.h`.

2.25.2.12 `template<typename _BI1, typename _BI2> _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result) [inline]`

Moves the range [first,last) into result.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

Returns

`result - (first - last)`

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range (first,last]. Use `move` instead. Note that the start of the output range may overlap [first,last).

Definition at line 655 of file `stl_algobase.h`.

2.25.2.13 `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred) [inline]`

Move elements for which a predicate is true to the beginning of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[__first,middle)` and false for each `i` in the range `[middle,__last)`.

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 5274 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__partition()`.

2.25.2.14 `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate > pair<_OutputIterator1, _OutputIterator2> std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__out_true</code>	An output iterator.
<code>__out_false</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `out_true` and each element for which `__pred` returns false to `__out_false`.

Definition at line 1066 of file `stl_algo.h`.

2.25.2.15 `template<typename _ForwardIterator, typename _Predicate > _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

Find the partition point of a partitioned range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__pred</code>	A predicate.

Returns

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

Definition at line 844 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.25.2.16 `template<typename _RandomAccessIterator > void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Randomly shuffle the elements of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

Nothing.

Reorder the elements in the range `[__first, __last)` using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 5210 of file `stl_algo.h`.

References `std::iter_swap()`.

2.25.2.17 `template<typename _RandomAccessIterator, typename _RandomNumberGenerator > void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`

Shuffle the elements of a sequence using a random number generator.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__rand</code>	The RNG functor or function.

Returns

Nothing.

Reorders the elements in the range `[__first, __last)` using `__rand` to provide a random distribution. Calling `__rand(-N)` for a positive integer `N` should return a randomly chosen integer from the range `[0,N)`.

Definition at line 5238 of file `stl_algo.h`.

References `std::iter_swap()`.

2.25.2.18 `template<typename _ForwardIterator, typename _Tp > _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`

Remove elements from a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range `[__first,__last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1115 of file `stl_algo.h`.

2.25.2.19 `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::remove_copy (`
`_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`

Copy a sequence, removing elements of a given value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__value</code>	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` not equal to `__value` to the range beginning at `__result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 897 of file `stl_algo.h`.

2.25.2.20 `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::remove_copy_if`
`(_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`

Copy a sequence, removing elements for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 935 of file `stl_algo.h`.

2.25.2.21 `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

Remove elements from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range `[__first, __last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1158 of file `stl_algo.h`.

2.25.2.22 `template<typename _ForwardIterator, typename _Tp> void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __old_value, const _Tp & __new_value)`

Replace each occurrence of one value in a sequence with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

Returns

`replace()` returns no value.

For each iterator `i` in the range `[__first, __last)` if `*i == __old_value` then the assignment `*i = __new_value` is performed.

Definition at line 5007 of file `stl_algo.h`.

2.25.2.23 `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp> _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`

Copy a sequence, replacing each value for which a predicate returns true with another value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

Returns

The end of the output sequence, `__result+(__last-__first)`.

Copies each element in the range `[__first,__last)` to the range `[__result,__result+(__last-__first))` replacing elements for which `__pred` returns true with `__new_value`.

Definition at line 3945 of file `stl_algo.h`.

2.25.2.24 `template<typename _ForwardIterator, typename _Predicate, typename _Tp> void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`

Replace each value in a sequence for which a predicate returns true with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

Returns

`replace_if()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `__pred(*i)` is true then the assignment `*i = __new_value` is performed.

Definition at line 5039 of file `stl_algo.h`.

2.25.2.25 `template<typename _BidirectionalIterator> void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last) [inline]`

Reverse a sequence.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.

Returns

`reverse()` returns no value.

Reverses the order of the elements in the range `[__first,__last)`, so that the first element becomes the last etc. For every `i` such that $0 \leq i < (_last - _first)/2$, `reverse()` swaps `*(__first+i)` and `*(__last-(i+1))`

Definition at line 1466 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

2.25.2.26 `template<typename _BidirectionalIterator, typename _OutputIterator> _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`

Copy a sequence, reversing its elements.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[__first, __last)` to the range `[__result, __result+(__last-__first))` such that the order of the elements is reversed. For every `i` such that $0 \leq i < (__last - __first)$, `reverse_copy()` performs the assignment `*(__result+(__last-__first)-1-i) = *(__first+i)`. The ranges `[__first, __last)` and `[__result, __result+(__last-__first))` must not overlap.

Definition at line 1493 of file `stl_algo.h`.

```
2.25.2.27 template<typename _ForwardIterator > void std::rotate ( _ForwardIterator __first, _ForwardIterator __middle,
    _ForwardIterator __last ) [inline]
```

Rotate the elements of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

Nothing.

Rotates the elements of the range `[__first, __last)` by `(__middle - __first)` positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range `[__first, __last)`.

This effectively swaps the ranges `[__first, __middle)` and `[__middle, __last)`.

Performs `*(__first+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0, __last-__first)`.

Definition at line 1699 of file `stl_algo.h`.

References `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__stable_partition_adaptive()`.

```
2.25.2.28 template<typename _ForwardIterator, typename _OutputIterator > _OutputIterator std::rotate_copy ( _ForwardIterator
    __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result )
```

Copy a sequence, rotating its elements.

Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[__first,__last)` to the range beginning at

Returns

, rotating the copied elements by `(__middle-__first)` positions so that the element at `__middle` is moved to `__result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range `[__first,__last)`.

Performs `*(__result+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0,__last-__first)`.

Definition at line 1735 of file `stl_algo.h`.

2.25.2.29 `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator > void std::shuffle (`
`_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`

Shuffle the elements of a sequence using a uniform random number generator.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__g</code>	A <code>UniformRandomNumberGenerator</code> (26.5.1.3).

Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__g` to provide random numbers.

Definition at line 4390 of file `stl_algo.h`.

References `std::iter_swap()`.

2.25.2.30 `template<typename _ForwardIterator, typename _Predicate > _ForwardIterator std::stable_partition (`
`_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__pred(x) == __pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1914 of file `stl_algo.h`.

References `std::__find_if_not()`, `std::__inplace_stable_partition()`, `std::__stable_partition_adaptive()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::requested_size()`, and `std::Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.25.2.31 `template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator2 std::swap_ranges (`
`_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

Swap the elements of two sequences.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.

Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 165 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

2.25.2.32 `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation > _OutputIterator`
`std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`

Perform an operation on a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__unary_op</code>	A unary operator.

Returns

An output iterator equal to `__result+(__last-__first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=unary_op(*(__first+N))` for each `N` in the range `[0, __last-__first)`.

`unary_op` must not alter its argument.

Definition at line 4938 of file `stl_algo.h`.

2.25.2.33 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator`
`__result, _BinaryOperation __binary_op)`

Perform an operation on corresponding elements of two sequences.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_op</code>	A binary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=__binary_op(*(__first1+N),*(__first2+N))` for each `N` in the range `[0,__last1-__first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4975 of file `stl_algo.h`.

2.25.2.34 `template<typename _ForwardIterator > _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`

Remove consecutive duplicate values from a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1198 of file `stl_algo.h`.

2.25.2.35 `template<typename _ForwardIterator, typename _BinaryPredicate > _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`

Remove consecutive values from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1238 of file `stl_algo.h`.

2.25.2.36 `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result) [inline]`

Copy a sequence, removing consecutive duplicate values.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first, __last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 5139 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.25.2.37 `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred) [inline]`

Copy a sequence, removing consecutive values using a predicate.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first, __last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 5179 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.26 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits`
`< _InputIterator >`
`::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits`
`< _InputIterator >`
`::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

2.26.1 Detailed Description

2.26.2 Function Documentation

2.26.2.1 `template<typename _ForwardIterator > _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`

Find two adjacent values in a sequence that are equal.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first, __last)` and such that `*i == *(i+1)`, or `__last` if no such iterator exists.

Definition at line 4581 of file `stl_algo.h`.

2.26.2.2 `template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`

Find two adjacent values in a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first, __last)` and such that `__binary_pred(*i, *(i+1))` is true, or `__last` if no such iterator exists.

Definition at line 4613 of file `stl_algo.h`.

2.26.2.3 `template<typename _InputIterator, typename _Predicate> bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Checks that a predicate is true for all the elements of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `__pred` is true for each element in the range `[__first, __last)`, and false otherwise.

Definition at line 753 of file `stl_algo.h`.

References `std::find_if_not()`.

2.26.2.4 `template<typename _InputIterator, typename _Predicate> bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Checks that a predicate is false for at least an element of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range `[__first, __last)` such that `__pred` is true, and false otherwise.

Definition at line 788 of file `stl_algo.h`.

References `std::none_of()`.

2.26.2.5 `template<typename _InputIterator, typename _Tp> iterator_traits<_InputIterator>::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp & __value)`

Count the number of copies of a value in a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be counted.

Returns

The number of iterators `i` in the range `[__first, __last)` for which `*i == __value`

Definition at line 4645 of file `stl_algo.h`.

2.26.2.6 `template<typename _InputIterator, typename _Predicate> iterator_traits<_InputIterator>::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

Count the elements of a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The number of iterators `i` in the range `[__first, __last)` for which `__pred(*i)` is true.

Definition at line 4670 of file `stl_algo.h`.

Referenced by `std::is_permutation()`.

2.26.2.7 `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate> bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred) [inline]`

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1053 of file `stl_algobase.h`.

2.26.2.8 `template<typename _I1, typename _I2> bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2) [inline]`

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1021 of file `stl_algobase.h`.

Referenced by `std::operator==()`.

2.26.2.9 `template<typename _InputIterator, typename _Tp> _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp & __val) [inline]`

Find the first occurrence of a value in a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__val</code>	The value to find.

Returns

The first iterator `i` in the range `[__first,__last)` such that `*i == __val`, or `__last` if no such iterator exists.

Definition at line 4455 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

2.26.2.10 `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2) [inline]`

Find last matching subsequence in a sequence.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.

Returns

The last iterator *i* in the range `[__first1, __last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each *N* in the range `[0, __last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence. This means that the returned iterator *i* will be in the range `[__first1, __last1-(__last2-__first2))`

Definition at line 671 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.26.2.11 template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1
std::find_end( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2,
               _BinaryPredicate __comp ) [inline]
```

Find last matching subsequence in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.
<code>__comp</code>	The predicate to use.

Returns

The last iterator *i* in the range `[__first1, __last1-(__last2-__first2))` such that `predicate(*(i+N), (__first2+N))` is true for each *N* in the range `[0, __last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence. This means that the returned iterator *i* will be in the range `[__first1, __last1-(__last2-__first2))`

Definition at line 719 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.26.2.12 template<typename _InputIterator, typename _ForwardIterator > _InputIterator std::find_first_of( _InputIterator __first1,
                                                       _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2 )
```

Find element from a set in a sequence.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.

Returns

The first iterator `i` in the range `[__first1, __last1)` such that `*i == *(i2)` such that `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 4509 of file `stl_algo.h`.

```
2.26.2.13 template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _InputIterator
std::find_first_of ( _InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2,
    _BinaryPredicate __comp )
```

Find element from a set in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.
<code>__comp</code>	Predicate to use.

Returns

The first iterator `i` in the range `[__first1, __last1)` such that `comp(*i, *(i2))` is true and `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 4550 of file `stl_algo.h`.

```
2.26.2.14 template<typename _InputIterator, typename _Predicate> _InputIterator std::find_if ( _InputIterator __first, _InputIterator
    __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The first iterator `i` in the range `[__first, __last)` such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 4479 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

```
2.26.2.15 template<typename _InputIterator, typename _Predicate> _InputIterator std::find_if_not ( _InputIterator __first,
    _InputIterator __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is false.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The first iterator `i` in the range `[__first, __last)` such that `__pred(*i)` is false, or `__last` if no such iterator exists.

Definition at line 803 of file `stl_algo.h`.

References `std::__find_if_not()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

2.26.2.16 `template<typename _InputIterator, typename _Function> _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`

Apply a function to every element of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__f</code>	A unary function object.

Returns

`__f` (`std::move(__f)` in C++0x).

Applies the function object `__f` to each element in the range `[first, last)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

Definition at line 4434 of file `stl_algo.h`.

2.26.2.17 `template<typename _ForwardIterator1, typename _ForwardIterator2> bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

Checks whether a permutaion of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 4294 of file `stl_algo.h`.

References `std::advance()`, `std::count()`, and `std::distance()`.

2.26.2.18 `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> bool
std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate
__pred)`

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__pred</code>	A binary predicate.

Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `_ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 4340 of file `stl_algo.h`.

References `std::advance()`, `std::bind()`, `std::count_if()`, and `std::distance()`.

2.26.2.19 `template<typename _InputIterator1, typename _InputIterator2> pair<_InputIterator1, _InputIterator2> std::mismatch (`
`_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1160 of file `stl_algobase.h`.

2.26.2.20 `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate> pair<_InputIterator1,`
`_InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate`
`__binary_pred)`

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the binary `_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1198 of file `stl_algobase.h`.

2.26.2.21 `template<typename _InputIterator, typename _Predicate> bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Checks that a predicate is false for all the elements of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range `[__first, __last)`, and false otherwise.

Definition at line 770 of file `stl_algo.h`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

2.26.2.22 `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`

Search a sequence for a matching sub-sequence.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.

Returns

The first iterator `i` in the range `[__first1, __last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range `[__first1, __last1-(__last2-__first2))`

Definition at line 4712 of file `stl_algo.h`.

2.26.2.23 `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1
std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2,
_BinaryPredicate __predicate)`

Search a sequence for a matching sub-sequence using a predicate.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.
<code>__predicate</code>	A binary predicate.

Returns

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `__predicate(*(i+N), *(__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)`, using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

See Also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4784 of file `stl_algo.h`.

2.26.2.24 `template<typename _ForwardIterator, typename _Integer, typename _Tp > _ForwardIterator std::search_n (`
`_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val)`

Search a sequence for a number of consecutive values.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.

Returns

The first iterator `i` in the range `[__first, __last - __count)` such that `*(i+N) == __val` for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `count` consecutive elements equal to `__val`.

Definition at line 4858 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

2.26.2.25 `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate > _ForwardIterator
std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred)`

Search a sequence for a number of consecutive values using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.
<code>__binary_pred</code>	A binary predicate.

Returns

The first iterator `i` in the range `[__first, __last-__count)` such that `__binary_pred(*(i+N), __val)` is true for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

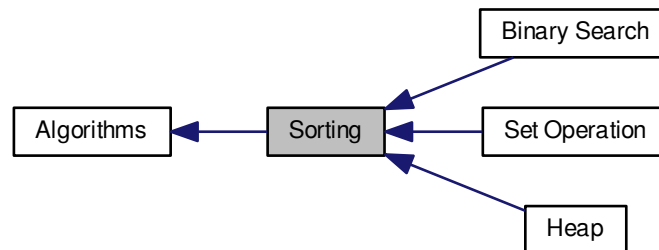
Searches the range `[__first, __last)` for `__count` consecutive elements for which the predicate returns true.

Definition at line 4896 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

2.27 Sorting

Collaboration diagram for Sorting:



Modules

- [Binary Search](#)
- [Heap](#)
- [Set Operation](#)

Functions

- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

2.27.1 Detailed Description

2.27.2 Function Documentation

2.27.2.1 `template<typename _BidirectionalIterator > void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`

Merges two sorted ranges in place.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`, `__middle`) and [`__middle`, `__last`), and puts the result in [`__first`, `__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes $(\text{__last} - \text{__first}) - 1$ comparisons. Otherwise an $N \log N$ algorithm is used, where N is `distance(__first, __last)`.

Definition at line 3177 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::distance()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.27.2.2 `template<typename _BidirectionalIterator, typename _Compare > void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`

Merges two sorted ranges in place.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A functor to use for comparisons.

Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`middle`,`last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes $(\text{__last} - \text{__first}) - 1$ comparisons. Otherwise an $N \log N$ algorithm is used, where N is `distance(__first, __last)`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 3232 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::distance()`, and `std::Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.27.2.3 `template<typename _ForwardIterator> bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
`[inline]`

Determines whether the elements of a sequence are sorted.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3975 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.27.2.4 `template<typename _ForwardIterator, typename _Compare> bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)` `[inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3989 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.27.2.5 `template<typename _ForwardIterator> _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`

Determines the end of a sorted sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 4003 of file `stl_algo.h`.

2.27.2.6 `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

Determines the end of a sorted sequence using comparison functor.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 4032 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

2.27.2.7 `template<typename _I1, typename _I2> bool std::lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2) [inline]`

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

A boolean true or false.

Returns true if the sequence of elements defined by the range `[first1,last1)` is lexicographically less than the sequence of elements defined by the range `[first2,last2)`. Returns false otherwise. (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1084 of file `stl_algobase.h`.

2.27.2.8 `template<typename _I1, typename _I2, typename _Compare> bool std::lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A comparison functor .

Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1120 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

2.27.2.9 `template<typename _Tp> const _Tp & std::max (const _Tp & __a, const _Tp & __b) [inline]`

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 216 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_nth_element()`, `std::_Deque_base<_Tp, _Alloc>::_M_initialize_map()`, `std::deque<_Tp, _Alloc>::_M_reallocate_map()`, `std::discard_block_engine<_RandomNumberEngine, __p, __r>::max()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::max()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`.

2.27.2.10 `template<typename _Tp, typename _Compare> const _Tp & std::max (const _Tp & __a, const _Tp & __b, _Compare __comp) [inline]`

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 260 of file `stl_algobase.h`.

2.27.2.11 `template<typename _ForwardIterator > _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`

Return the maximum element in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

Iterator referencing the first instance of the largest value.

Definition at line 6298 of file `stl_algo.h`.

2.27.2.12 `template<typename _ForwardIterator, typename _Compare > _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

Return the maximum element in a range using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

Iterator referencing the first instance of the largest value according to `__comp`.

Definition at line 6326 of file `stl_algo.h`.

Referenced by `std::valarray<_Tp>::max()`.

2.27.2.13 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.

Returns

An iterator pointing to the first element *not less than* *val*.

Merges the ranges `[__first1,__last1)` and `[__first2,__last2)` into the sorted range `[__result, __result + (__last1-__first1) + (__last2-__first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input

ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5540 of file `stl_algo.h`.

```
2.27.2.14 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
    _OutputIterator std::merge ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,
        _OutputIterator __result, _Compare __comp )
```

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5604 of file `stl_algo.h`.

```
2.27.2.15 template<typename _Tp> const _Tp & std::min ( const _Tp & __a, const _Tp & __b ) [inline]
```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 193 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::parallel_random_shuffle_drs()`, `__gnu_parallel::parallel_sort_qs_divide()`, `__gnu_profile::report()`, `__gnu_parallel::search_template()`, `__gnu_parallel::sequential_random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `std::basic_string<char>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::generate_canonical()`, `std::discard_block_engine<_RandomNumberEngine, __p, __r>::min()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::min()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`,

`__gnu_cxx::random_sample_n()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `__gnu_cxx::__versa_string< -_CharT, _Traits, _Alloc, _Base >::rfind()`, `std::basic_string< _CharT, _Traits, _Alloc >::rfind()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, `std::basic_filebuf< _CharT, _Traits >::xsputn()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

2.27.2.16 `template<typename _Tp, typename _Compare> const _Tp & std::min (const _Tp & __a, const _Tp & __b, _Compare __comp) [inline]`

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 239 of file `stl_algo.h`.

2.27.2.17 `template<typename _ForwardIterator> _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`

Return the minimum element in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

Iterator referencing the first instance of the smallest value.

Definition at line 6242 of file `stl_algo.h`.

2.27.2.18 `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

Return the minimum element in a range using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

Iterator referencing the first instance of the smallest value according to `__comp`.

Definition at line 6270 of file `stl_algo.h`.

Referenced by `std::valarray<_Tp>::min()`.

2.27.2.19 `template<typename _Tp> pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b)`
`[inline]`

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 4062 of file `stl_algo.h`.

2.27.2.20 `template<typename _Tp, typename _Compare> pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)` `[inline]`

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 4082 of file `stl_algo.h`.

2.27.2.21 `template<typename _ForwardIterator> pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

make_pair(m, M), where m is the first iterator i in [`__first`, `__last`) such that no other element in the range is smaller, and where M is the last iterator i in [`__first`, `__last`) such that no other element in the range is larger.

Definition at line 4101 of file `stl_algo.h`.

References `std::make_pair()`.

2.27.2.22 `template<typename _ForwardIterator, typename _Compare> pair<_ForwardIterator, _ForwardIterator>
std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

`make_pair(m, M)`, where `m` is the first iterator `i` in `[__first, __last)` such that no other element in the range is smaller, and where `M` is the last iterator `i` in `[__first, __last)` such that no other element in the range is larger.

Definition at line 4177 of file `stl_algo.h`.

References `std::make_pair()`.

2.27.2.23 `template<typename _BidirectionalIterator> bool std::next_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last)`

Permute range into the next *dictionary* ordering.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3677 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.27.2.24 `template<typename _BidirectionalIterator, typename _Compare> bool std::next_permutation (_BidirectionalIterator
__first, _BidirectionalIterator __last, _Compare __comp)`

Permute range into the next *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range `[__first,__last)` as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3734 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.27.2.25 `template<typename _RandomAccessIterator > void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last) [inline]`

Sort a sequence just enough to find a particular position.

Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `*j < *i` is false.

Definition at line 5384 of file `stl_algo.h`.

References `std::__lg()`.

2.27.2.26 `template<typename _RandomAccessIterator, typename _Compare > void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp) [inline]`

Sort a sequence just enough to find a particular position using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `__comp(*j,*i)` is false.

Definition at line 5423 of file `stl_algo.h`.

References `std::__lg()`.

2.27.2.27 `template<typename _RandomAccessIterator > void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last) [inline]`

Sort the smallest elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[__first,__last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*j<*i` and `*k<*i` are both false.

Definition at line 5308 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

2.27.2.28 `template<typename _RandomAccessIterator, typename _Compare > void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the smallest elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[__first,__last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*__comp(j,*i)` and `*__comp(*k,*i)` are both false.

Definition at line 5347 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

2.27.2.29 `template<typename _InputIterator, typename _RandomAccessIterator > _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`

Copy the smallest elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range $[_first, _last)$ to the range beginning at `__result_first`, where the number of elements to be copied, N , is the smaller of $(_last - _first)$ and $(_result_last - _result_first)$. After the sort if i and j are iterators in the range $[_result_first, _result_first + N)$ such that i precedes j then $*j < *i$ is false. The value returned is `__result_first + N`.

Definition at line 1998 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

```
2.27.2.30  template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator
            std::partial_sort_copy ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first,
                                   _RandomAccessIterator __result_last, _Compare __comp )
```

Copy the smallest elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	Another input iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.
<code>__comp</code>	A comparison functor.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range $[_first, _last)$ to the range beginning at `result_first`, where the number of elements to be copied, N , is the smaller of $(_last - _first)$ and $(_result_last - _result_first)$. After the sort if i and j are iterators in the range $[_result_first, _result_first + N)$ such that i precedes j then `__comp(*j, *i)` is false. The value returned is `__result_first + N`.

Definition at line 2064 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

```
2.27.2.31  template<typename _BidirectionalIterator> bool std::prev_permutation ( _BidirectionalIterator __first,
                                         _BidirectionalIterator __last )
```

Permute range into the previous *dictionary* ordering.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3790 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.27.2.32 `template<typename _BidirectionalIterator, typename _Compare> bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range `[__first,__last)` as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3847 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.27.2.33 `template<typename _RandomAccessIterator> void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Sort the elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator *i* in the range `[__first,__last-1)`, `*(i+1)< *i` is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5461 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

2.27.2.34 `template<typename _RandomAccessIterator, typename _Compare> void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that `__comp(*(i+1), *i)` is false for every iterator `i` in the range `[__first, __last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5497 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

2.27.2.35 `template<typename _RandomAccessIterator> void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Sort the elements of a sequence, preserving the relative order of equivalent elements.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that for each iterator `i` in the range `[__first, __last-1)`, `*(i+1) < *i` is false.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first, __last)` such that `x < y` is false and `y < x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5663 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::__Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::__Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.27.2.36 `template<typename _RandomAccessIterator, typename _Compare> void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator `i` in the range `[__first,__last-1)`, `__comp(*(i+1),*i)` is false.

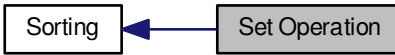
The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__comp(x,y)` is false and `__comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5705 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::Temporary_buffer<_ForwardIterator, _Tp >::begin()`, and `std::Temporary_buffer<_ForwardIterator, _Tp >::size()`.

2.28 Set Operation

Collaboration diagram for Set Operation:



Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

2.28.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

2.28.2 Function Documentation

2.28.2.1 `template<typename _InputIterator1, typename _InputIterator2 > bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

Determines whether all elements of a sequence exists in a range.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.

Returns

True if each element in [`__first2`,`__last2`) is contained in order within [`__first1`,`__last1`). False otherwise.

This operation expects both [`__first1`,`__last1`) and [`__first2`,`__last2`) to be sorted. Searches for the presence of each element in [`__first2`,`__last2`) within [`__first1`,`__last1`). The iterators over each range only move forward, so this is a linear algorithm. If an element in [`__first2`,`__last2`) is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 3572 of file `stl_algo.h`.

2.28.2.2 `template<typename _InputIterator1, typename _InputIterator2, typename _Compare > bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

Returns

True if each element in [`__first2`,`__last2`) is contained in order within [`__first1`,`__last1`) according to `comp`. False otherwise.

This operation expects both [`__first1`,`__last1`) and [`__first2`,`__last2`) to be sorted. Searches for the presence of each element in [`__first2`,`__last2`) within [`__first1`,`__last1`), using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [`__first2`,`__last2`) is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 3623 of file `stl_algo.h`.

2.28.2.3 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the difference of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6001 of file `stl_algo.h`.

```
2.28.2.4 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
        _OutputIterator std::set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
        __last2, _OutputIterator __result, _Compare __comp )
```

Return the difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6062 of file `stl_algo.h`.

```
2.28.2.5 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
        std::set_intersection ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,
        _OutputIterator __result )
```

Return the intersection of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5886 of file `stl_algo.h`.

```
2.28.2.6 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
        _OutputIterator std::set_intersection ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
        __last2, _OutputIterator __result, _Compare __comp )
```

Return the intersection of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5943 of file `stl_algo.h`.

```
2.28.2.7 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
        std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
        __last2, _OutputIterator __result )
```

Return the symmetric difference of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6120 of file `stl_algo.h`.

```
2.28.2.8 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
    _OutputIterator std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
    _InputIterator2 __last2, _OutputIterator __result, _Compare __comp )
```

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6186 of file `stl_algo.h`.

```
2.28.2.9 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_union (
    _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result )
```

Return the union of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5752 of file `stl_algo.h`.

```
2.28.2.10 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
    _OutputIterator std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
    __last2, _OutputIterator __result, _Compare __comp )
```

Return the union of two sorted ranges using a comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5819 of file `stl_algo.h`.

2.29 Binary Search

Collaboration diagram for Binary Search:



Functions

- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator,`
`_ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator,`
`_ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _`
`Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _`
`Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _`
`Compare __comp)`

2.29.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

2.29.2 Function Documentation

2.29.2.1 `template<typename _ForwardIterator, typename _Tp> bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

True if `__val` (or its equivalent) is in `[__first, __last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2720 of file `stl_algo.h`.

References `std::lower_bound()`.

2.29.2.2 `template<typename _ForwardIterator, typename _Tp, typename _Compare> bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

True if `__val` (or its equivalent) is in `[__first, __last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2753 of file `stl_algo.h`.

References `std::lower_bound()`.

2.29.2.3 `template<typename _ForwardIterator, typename _Tp> pair<_ForwardIterator, _ForwardIterator> std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

Definition at line 2597 of file stl_algo.h.

References std::advance(), std::distance(), std::lower_bound(), and std::upper_bound().

2.29.2.4 `template<typename _ForwardIterator, typename _Tp, typename _Compare> pair<_ForwardIterator, _ForwardIterator> std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

Definition at line 2659 of file stl_algo.h.

References std::advance(), std::distance(), std::lower_bound(), and std::upper_bound().

2.29.2.5 `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`

Finds the first position in which `val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element *not less than val*, or end() if every element is less than *val*.

Definition at line 943 of file stl_algobase.h.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

2.29.2.6 `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::lower_bound (`
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Finds the first position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than* `__val`, or `end()` if every element is less than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2448 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.29.2.7 `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::upper_bound (`
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

Definition at line 2495 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.29.2.8 `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::upper_bound (`
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2544 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, and `std::equal_range()`.

2.30 Allocators

Collaboration diagram for Allocators:



Classes

- struct `__gnu_cxx::__alloc_traits< _Alloc >`
Uniform interface to C++98 and C++0x allocators.
- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`
This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.
- class `__gnu_cxx::__pool_alloc< _Tp >`
Allocator using a memory pool with a single lock.
- class `__gnu_cxx::__ExtPtr_allocator< _Tp >`
An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.
- class `__gnu_cxx::array_allocator< _Tp, _Array >`
An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class `__gnu_cxx::bitmap_allocator< _Tp >`
Bitmap Allocator, primary template.
- class `__gnu_cxx::debug_allocator< _Alloc >`
A meta-allocator with debugging bits, as per [20.4].
This is precisely the allocator defined in the C++ Standard.
- class `__gnu_cxx::malloc_allocator< _Tp >`
An allocator that uses `malloc`.
This is precisely the allocator defined in the C++ Standard.
- class `__gnu_cxx::new_allocator< _Tp >`
An allocator that uses global `new`, as per [20.4].
This is precisely the allocator defined in the C++ Standard.
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`
Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.
Note: Deallocate not allowed to throw.
- class `std::allocator< void >`
`allocator<void>` specialization.
- struct `std::allocator_traits< _Alloc >`
Uniform interface to all allocator types.
- struct `uses_allocator< typename, typename >`
Declare `uses_allocator` so it can be specialized in `<queue>` etc.

Typedefs

- `template<typename _Tp >`
`using std::__allocator_base = __gnu_cxx::new_allocator<_Tp >`

Functions

- `template<typename _Alloc >`
`auto std::__do_outmost (_Alloc &__a, _Alloc *) -> decltype(__a.outer_allocator())`
- `template<typename _Alloc >`
`_Alloc & std::__do_outmost (_Alloc &__a,...)`
- `template<typename _Alloc >`
`auto std::__outmost (_Alloc &__a) -> decltype(__do_outmost(__a,&__a))`
- `template<typename _T1 , typename _T2 >`
`bool std::operator!= (const allocator<_T1 > &, const allocator<_T2 > &)`
- `template<typename _Tp >`
`bool std::operator!= (const allocator<_Tp > &, const allocator<_Tp > &)`
- `template<typename _OutA1 , typename _OutA2 , typename... _InA>`
`bool std::operator!= (const scoped_allocator_adaptor<_OutA1, _InA...> &__a, const scoped_allocator_adaptor<_OutA2, _InA...> &__b) noexcept`
- `template<typename _T1 , typename _T2 >`
`bool std::operator== (const allocator<_T1 > &, const allocator<_T2 > &)`
- `template<typename _Tp >`
`bool std::operator== (const allocator<_Tp > &, const allocator<_Tp > &)`
- `template<typename _OutA1 , typename _OutA2 , typename... _InA>`
`bool std::operator== (const scoped_allocator_adaptor<_OutA1, _InA...> &__a, const scoped_allocator_adaptor<_OutA2, _InA...> &__b) noexcept`

2.30.1 Detailed Description

Classes encapsulating memory operations.

2.30.2 Typedef Documentation

2.30.2.1 `template<typename _Tp > using std::__allocator_base = typedef __gnu_cxx::new_allocator<_Tp>`

An alias to the base class for `std::allocator`.

Used to set the `std::allocator` base class to `__gnu_cxx::new_allocator`.

Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 48 of file `c++/allocator.h`.

2.31 Atomics

Classes

- struct `std::__atomic_base<_PTp*>`
Partial specialization for pointer types.
- struct `std::__atomic_flag_base`
Base type for `atomic_flag`.
- struct `std::atomic<_Tp>`
Generic atomic type, primary class template.
- struct `std::atomic<_Tp*>`
Partial specialization for pointer types.
- struct `std::atomic<bool>`
Explicit specialization for `bool`.
- struct `std::atomic<char>`
Explicit specialization for `char`.
- struct `std::atomic<char16_t>`
Explicit specialization for `char16_t`.
- struct `std::atomic<char32_t>`
Explicit specialization for `char32_t`.
- struct `std::atomic<int>`
Explicit specialization for `int`.
- struct `std::atomic<long>`
Explicit specialization for `long`.
- struct `std::atomic<long long>`
Explicit specialization for `long long`.
- struct `std::atomic<short>`
Explicit specialization for `short`.
- struct `std::atomic<signed char>`
Explicit specialization for `signed char`.
- struct `std::atomic<unsigned char>`
Explicit specialization for `unsigned char`.
- struct `std::atomic<unsigned int>`
Explicit specialization for `unsigned int`.
- struct `std::atomic<unsigned long>`
Explicit specialization for `unsigned long`.
- struct `std::atomic<unsigned long long>`
Explicit specialization for `unsigned long long`.
- struct `std::atomic<unsigned short>`
Explicit specialization for `unsigned short`.
- struct `std::atomic<wchar_t>`
Explicit specialization for `wchar_t`.
- struct `std::atomic_bool`
`atomic_bool`
- struct `std::atomic_flag`
`atomic_flag`

Macros

- `#define ATOMIC_BOOL_LOCK_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_VAR_INIT(_VI)`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

Typedefs

- `typedef unsigned char std::atomic_flag_data_type`
- `typedef __atomic_base< char > std::atomic_char`
- `typedef __atomic_base< char16_t > std::atomic_char16_t`
- `typedef __atomic_base< char32_t > std::atomic_char32_t`
- `typedef __atomic_base< int > std::atomic_int`
- `typedef __atomic_base< int_fast16_t > std::atomic_int_fast16_t`
- `typedef __atomic_base< int_fast32_t > std::atomic_int_fast32_t`
- `typedef __atomic_base< int_fast64_t > std::atomic_int_fast64_t`
- `typedef __atomic_base< int_fast8_t > std::atomic_int_fast8_t`
- `typedef __atomic_base< int_least16_t > std::atomic_int_least16_t`
- `typedef __atomic_base< int_least32_t > std::atomic_int_least32_t`
- `typedef __atomic_base< int_least64_t > std::atomic_int_least64_t`
- `typedef __atomic_base< int_least8_t > std::atomic_int_least8_t`
- `typedef __atomic_base< intmax_t > std::atomic_intmax_t`
- `typedef __atomic_base< intptr_t > std::atomic_intptr_t`
- `typedef __atomic_base< long long > std::atomic_llong`
- `typedef __atomic_base< long > std::atomic_long`
- `typedef __atomic_base< ptrdiff_t > std::atomic_ptrdiff_t`
- `typedef __atomic_base< signed char > std::atomic_schar`
- `typedef __atomic_base< short > std::atomic_short`
- `typedef __atomic_base< size_t > std::atomic_size_t`
- `typedef __atomic_base< unsigned char > std::atomic_uchar`
- `typedef __atomic_base< unsigned int > std::atomic_uint`

- typedef __atomic_base
< uint_fast16_t > [std::atomic_uint_fast16_t](#)
- typedef __atomic_base
< uint_fast32_t > [std::atomic_uint_fast32_t](#)
- typedef __atomic_base
< uint_fast64_t > [std::atomic_uint_fast64_t](#)
- typedef __atomic_base
< uint_fast8_t > [std::atomic_uint_fast8_t](#)
- typedef __atomic_base
< uint_least16_t > [std::atomic_uint_least16_t](#)
- typedef __atomic_base
< uint_least32_t > [std::atomic_uint_least32_t](#)
- typedef __atomic_base
< uint_least64_t > [std::atomic_uint_least64_t](#)
- typedef __atomic_base
< uint_least8_t > [std::atomic_uint_least8_t](#)
- typedef __atomic_base< uintmax_t > [std::atomic_uintmax_t](#)
- typedef __atomic_base< uintptr_t > [std::atomic_uintptr_t](#)
- typedef __atomic_base
< unsigned long long > [std::atomic_ullong](#)
- typedef __atomic_base
< unsigned long > [std::atomic_ulong](#)
- typedef __atomic_base
< unsigned short > [std::atomic_ushort](#)
- typedef __atomic_base< wchar_t > [std::atomic_wchar_t](#)
- typedef enum [std::memory_order](#) [std::memory_order](#)

Enumerations

- enum __memory_order_modifier { __memory_order_mask, __memory_order_modifier_mask, __memory_order_hle_acquire, __memory_order_hle_release }
- enum [std::memory_order](#) {
 memory_order_relaxed, **memory_order_consume**, **memory_order_acquire**, **memory_order_release**,
 memory_order_acq_rel, **memory_order_seq_cst** }

Functions

- constexpr memory_order **std::__cmpexch_failure_order** (memory_order __m) noexcept
- constexpr memory_order **std::__cmpexch_failure_order2** (memory_order __m) noexcept
- template<typename _ITp >
 bool **std::atomic_compare_exchange_strong** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept
- template<typename _ITp >
 bool **std::atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept
- template<typename _ITp >
 bool **std::atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
 bool **std::atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
 bool **std::atomic_compare_exchange_weak** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept

- `template<typename _ITp >`
`bool std::atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_exchange (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_add (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_add (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_add_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_add_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`

- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_sub (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_sub (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_sub_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp * std::atomic_fetch_sub_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`
`void std::atomic_init (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`void std::atomic_init (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `void std::atomic_signal_fence (memory_order __m) noexcept`

- `template<typename _ITp >`
`void std::atomic_store (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`void std::atomic_store (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`
`void std::atomic_store_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`void std::atomic_store_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `void std::atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Tp >`
`_Tp std::kill_dependency (_Tp __y) noexcept`
- `constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod)`
- `constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod)`

2.31.1 Detailed Description

Components for performing atomic operations.

2.31.2 Macro Definition Documentation

2.31.2.1 `#define ATOMIC_BOOL_LOCK_FREE`

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

Definition at line 49 of file `atomic_lockfree_defines.h`.

2.31.3 Typedef Documentation

2.31.3.1 `typedef __atomic_base<char> std::atomic_char`

`atomic_char`

Definition at line 117 of file `atomic_base.h`.

2.31.3.2 `typedef __atomic_base<char16_t> std::atomic_char16_t`

`atomic_char16_t`

Definition at line 156 of file `atomic_base.h`.

2.31.3.3 `typedef __atomic_base< char32_t > std::atomic_char32_t`

`atomic_char32_t`

Definition at line 159 of file `atomic_base.h`.

2.31.3.4 `typedef __atomic_base<int> std::atomic_int`

`atomic_int`

Definition at line 135 of file `atomic_base.h`.

2.31.3.5 typedef __atomic_base<int_fast16_t> std::atomic_int_fast16_t

atomic_int_fast16_t

Definition at line 197 of file atomic_base.h.

2.31.3.6 typedef __atomic_base<int_fast32_t> std::atomic_int_fast32_t

atomic_int_fast32_t

Definition at line 203 of file atomic_base.h.

2.31.3.7 typedef __atomic_base<int_fast64_t> std::atomic_int_fast64_t

atomic_int_fast64_t

Definition at line 209 of file atomic_base.h.

2.31.3.8 typedef __atomic_base<int_fast8_t> std::atomic_int_fast8_t

atomic_int_fast8_t

Definition at line 191 of file atomic_base.h.

2.31.3.9 typedef __atomic_base<int_least16_t> std::atomic_int_least16_t

atomic_int_least16_t

Definition at line 172 of file atomic_base.h.

2.31.3.10 typedef __atomic_base<int_least32_t> std::atomic_int_least32_t

atomic_int_least32_t

Definition at line 178 of file atomic_base.h.

2.31.3.11 typedef __atomic_base<int_least64_t> std::atomic_int_least64_t

atomic_int_least64_t

Definition at line 184 of file atomic_base.h.

2.31.3.12 typedef __atomic_base<int_least8_t> std::atomic_int_least8_t

atomic_int_least8_t

Definition at line 166 of file atomic_base.h.

2.31.3.13 typedef __atomic_base<intmax_t> std::atomic_intmax_t

atomic_intmax_t

Definition at line 225 of file atomic_base.h.

2.31.3.14 typedef __atomic_base<intptr_t> std::atomic_intptr_t

atomic_intptr_t

Definition at line 216 of file atomic_base.h.

2.31.3.15 `typedef __atomic_base<long long> std::atomic_llong`

`atomic_llong`

Definition at line 147 of file `atomic_base.h`.

2.31.3.16 `typedef __atomic_base<long> std::atomic_long`

`atomic_long`

Definition at line 141 of file `atomic_base.h`.

2.31.3.17 `typedef __atomic_base<ptrdiff_t> std::atomic_ptrdiff_t`

`atomic_ptrdiff_t`

Definition at line 231 of file `atomic_base.h`.

2.31.3.18 `typedef __atomic_base<signed char> std::atomic_schar`

`atomic_schar`

Definition at line 123 of file `atomic_base.h`.

2.31.3.19 `typedef __atomic_base<short> std::atomic_short`

`atomic_short`

Definition at line 129 of file `atomic_base.h`.

2.31.3.20 `typedef __atomic_base<size_t> std::atomic_size_t`

`atomic_size_t`

Definition at line 222 of file `atomic_base.h`.

2.31.3.21 `typedef __atomic_base<unsigned char> std::atomic_uchar`

`atomic_uchar`

Definition at line 126 of file `atomic_base.h`.

2.31.3.22 `typedef __atomic_base<unsigned int> std::atomic_uint`

`atomic_uint`

Definition at line 138 of file `atomic_base.h`.

2.31.3.23 `typedef __atomic_base<uint_fast16_t> std::atomic_uint_fast16_t`

`atomic_uint_fast16_t`

Definition at line 200 of file `atomic_base.h`.

2.31.3.24 `typedef __atomic_base<uint_fast32_t> std::atomic_uint_fast32_t`

`atomic_uint_fast32_t`

Definition at line 206 of file `atomic_base.h`.

2.31.3.25 `typedef __atomic_base<uint_fast64_t> std::atomic_uint_fast64_t`

`atomic_uint_fast64_t`

Definition at line 212 of file `atomic_base.h`.

2.31.3.26 `typedef __atomic_base<uint_fast8_t> std::atomic_uint_fast8_t`

`atomic_uint_fast8_t`

Definition at line 194 of file `atomic_base.h`.

2.31.3.27 `typedef __atomic_base<uint_least16_t> std::atomic_uint_least16_t`

`atomic_uint_least16_t`

Definition at line 175 of file `atomic_base.h`.

2.31.3.28 `typedef __atomic_base<uint_least32_t> std::atomic_uint_least32_t`

`atomic_uint_least32_t`

Definition at line 181 of file `atomic_base.h`.

2.31.3.29 `typedef __atomic_base<uint_least64_t> std::atomic_uint_least64_t`

`atomic_uint_least64_t`

Definition at line 187 of file `atomic_base.h`.

2.31.3.30 `typedef __atomic_base<uint_least8_t> std::atomic_uint_least8_t`

`atomic_uint_least8_t`

Definition at line 169 of file `atomic_base.h`.

2.31.3.31 `typedef __atomic_base<uintmax_t> std::atomic_uintmax_t`

`atomic_uintmax_t`

Definition at line 228 of file `atomic_base.h`.

2.31.3.32 `typedef __atomic_base<uintptr_t> std::atomic_uintptr_t`

`atomic_uintptr_t`

Definition at line 219 of file `atomic_base.h`.

2.31.3.33 `typedef __atomic_base<unsigned long long> std::atomic_ullong`

`atomic_ullong`

Definition at line 150 of file `atomic_base.h`.

2.31.3.34 `typedef __atomic_base<unsigned long> std::atomic_ulong`

`atomic_ulong`

Definition at line 144 of file `atomic_base.h`.

2.31.3.35 `typedef __atomic_base<unsigned short> std::atomic_ushort`

`atomic_ushort`

Definition at line 132 of file `atomic_base.h`.

2.31.3.36 `typedef __atomic_base<wchar_t> std::atomic_wchar_t`

`atomic_wchar_t`

Definition at line 153 of file `atomic_base.h`.

2.31.3.37 `typedef enum std::memory_order std::memory_order`

Enumeration for `memory_order`.

2.31.4 Enumeration Type Documentation

2.31.4.1 `enum std::memory_order`

Enumeration for `memory_order`.

Definition at line 52 of file `atomic_base.h`.

2.31.5 Function Documentation

2.31.5.1 `template<typename _Tp> _Tp std::kill_dependency (_Tp __y) [inline], [noexcept]`

`kill_dependency`

Definition at line 108 of file `atomic_base.h`.

2.32 Hashes

Collaboration diagram for Hashes:



Classes

- struct `hash<_Tp>`
Primary class template hash.
- struct `std::hash<_Tp*>`
Partial specializations for pointer types.
- struct `std::hash<bool>`
Explicit specialization for bool.
- struct `std::hash<char>`
Explicit specialization for char.
- struct `std::hash<char16_t>`
Explicit specialization for char16_t.
- struct `std::hash<char32_t>`
Explicit specialization for char32_t.
- struct `std::hash<double>`
Specialization for double.
- struct `std::hash<float>`
Specialization for float.
- struct `std::hash<int>`
Explicit specialization for int.
- struct `std::hash<long>`
Explicit specialization for long.
- struct `std::hash<long double>`
Specialization for long double.
- struct `std::hash<long long>`
Explicit specialization for long long.
- struct `std::hash<short>`
Explicit specialization for short.
- struct `std::hash<signed char>`
Explicit specialization for signed char.
- struct `std::hash<unsigned char>`
Explicit specialization for unsigned char.
- struct `std::hash<unsigned int>`
Explicit specialization for unsigned int.

- struct `std::hash< unsigned long >`
Explicit specialization for unsigned long.
- struct `std::hash< unsigned long long >`
Explicit specialization for unsigned long long.
- struct `std::hash< unsigned short >`
Explicit specialization for unsigned short.
- struct `std::hash< wchar_t >`
Explicit specialization for wchar_t.

Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

2.32.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

2.33 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- `struct _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >`
- `struct _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- `struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- `struct _Hash_node< _Value, _Cache_hash_code >`
- `struct _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >`
- `struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys >`
- `struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- `struct _Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- `struct std::__detail::_Before_begin< _NodeAlloc >`
- `struct std::__detail::_Default_ranged_hash`

Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that $h(k, N) = h2(h1(k), N)$, but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.

- `struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >`
Specialization.
- `struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >`
Specialization.
- `struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
Specialization.
- `struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
Specialization.
- `struct std::__detail::_Equality_base`
- `struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >`
Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.
- `struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`
Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.
- `struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

- struct `std::__detail::_Hash_node< _Value, false >`
- struct `std::__detail::_Hash_node< _Value, true >`
- struct `std::__detail::_Hash_node_base`
- struct `std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >`

Specialization not using EBO.

- struct `std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >`

Specialization using EBO.

- struct `std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >`
- struct `std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >`

Specialization.

- struct `std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >`

Specialization.

- struct `std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >`

Specialization.

- struct `std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- struct `std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`

local const iterators

- struct `std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`

local iterators

- struct `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`

Specialization.

- struct `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`

Specialization.

- struct `std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- struct `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`

Partial specialization, __unique_keys set to false.

- struct `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`

Partial specialization, __unique_keys set to true.

- struct `std::__detail::_Mod_range_hashing`

Default range hashing function: use division to fold a large number into the range [0, N).

- struct `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >`

Node const iterators, used to iterate through all the hashtable.

- struct `std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >`

Node iterators, used to iterate through all the hashtable.

- struct `std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >`

Base class for node iterators.

- struct `std::__detail::_Prime_rehash_policy`

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

- struct `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime, _rehash_policy, _Traits >`
Specialization.
- class `std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Functions

- template<class _Iterator >
`std::iterator_traits`
`< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- template<class _Iterator >
`std::iterator_traits`
`< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- template<class _Iterator >
`std::iterator_traits`
`< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last)`
- bool `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::M_equal (const __hashtable &) const`
- bool `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >::M_equal (const __hashtable &) const`
- template<typename _Uiterator >
`static bool std::__detail::_Equality_base::S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`
- mapped_type & `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::at (const key_type &__k)`
- const mapped_type & `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::at (const key_type &__k) const`
- template<typename _InputIterator >
`void std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::insert (_InputIterator __first, _InputIterator __last)`
- template<typename _Value, bool _Cache_hash_code>
`bool std::__detail::_operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)`
- template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>
`bool std::__detail::_operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- template<typename _Value, bool _Cache_hash_code>
`bool std::__detail::_operator== (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)`
- template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>
`bool std::__detail::_operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- mapped_type & `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::operator[] (const key_type &__k)`
- mapped_type & `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::operator[] (key_type &&__k)`

2.33.1 Detailed Description

2.34 Locales

Classes

- class `std::codecvt< _InternT, _ExternT, _StateT >`
Primary class template codecvt.
NB: Generic, mostly useless implementation.
- class `std::ctype< _CharT >`
Primary class template ctype facet.
This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.
- class `std::ctype< char >`
The ctype<char> specialization.
This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.
- class `std::ctype< wchar_t >`
The ctype<wchar_t> specialization.
This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
- class `std::locale`
Container class for localization functionality.
The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.
- class `std::locale::facet`
Localization functionality base class.
The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.
- class `std::locale::id`
Facet ID class.
The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.
- class `std::messages< _CharT >`
Primary class template messages.
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.
- struct `std::messages_base`
Messages facet base class providing catalog typedef.
- class `std::money_base`
Money format ordering data.
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.
- class `std::money_get< _CharT, _Inlter >`
Primary class template money_get.
This facet encapsulates the code to parse and return a monetary amount from a string.
- class `std::money_put< _CharT, _Outlter >`
Primary class template money_put.
This facet encapsulates the code to format and output a monetary amount.
- class `std::moneypunct< _CharT, _Intl >`
Primary class template moneypunct.
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class `std::num_get<_CharT, _InIter >`
Primary class template num_get.
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
- class `std::num_put<_CharT, _OutIter >`
Primary class template num_put.
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.
- class `std::numpunct<_CharT >`
Primary class template numpunct.
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
- class `std::time_base`
Time format ordering data.
This class provides an enum representing different orderings of time: day, month, and year.
- class `std::time_get<_CharT, _InIter >`
Primary class template time_get.
This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.
- class `std::time_put<_CharT, _OutIter >`
Primary class template time_put.
This facet encapsulates the code to format and output dates and times according to formats used by strftime().

Functions

- `template<typename _Facet >`
`bool std::has_facet (const locale & __loc) throw ()`
- `template<typename _Facet >`
`const _Facet & std::use_facet (const locale & __loc)`

2.34.1 Detailed Description

Classes and functions for internationalization and localization.

2.34.2 Function Documentation

2.34.2.1 `template<typename _Facet > bool std::has_facet (const locale & __loc) throw ()`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

Parameters

<code>__loc</code>	The locale to test.
--------------------	---------------------

Returns

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file `locale_classes.tcc`.

2.34.2.2 `template<typename _Facet> const _Facet & std::use_facet (const locale & __loc)`

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

Parameters

<code>__loc</code>	The locale to use.
--------------------	--------------------

Returns

Reference to facet of type `Facet`.

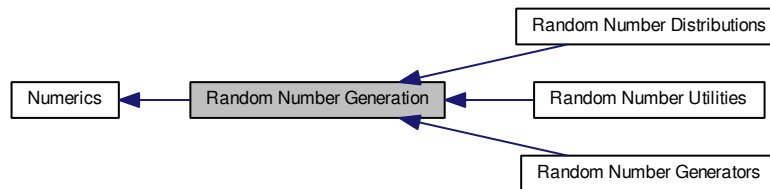
Exceptions

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
----------------------------	---

Definition at line 132 of file `locale_classes.tcc`.

2.35 Random Number Generation

Collaboration diagram for Random Number Generation:



Modules

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

Namespaces

- namespace [std::__detail](#)

Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator & __g)`

2.35.1 Detailed Description

A facility for generating random numbers on selected distributions.

2.35.2 Function Documentation

2.35.2.1 `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator & __g)`

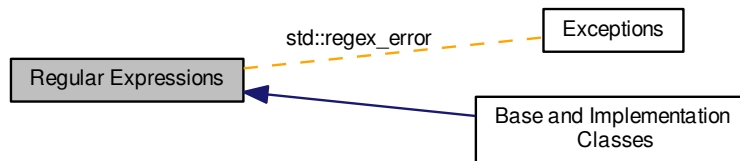
A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 3462 of file `bits/random.tcc`.

References `std::log()`, and `std::min()`.

2.36 Regular Expressions

Collaboration diagram for Regular Expressions:



Modules

- [Base and Implementation Classes](#)

Namespaces

- namespace [std::regex_constants](#)

Classes

- class [std::basic_regex< _Ch_type, _Rx_traits >](#)
- class [std::match_results< _Bi_iter, _Alloc >](#)
The results of a match or search operation.
- class [std::regex_error](#)
*A regular expression exception class.
 The regular expression library throws objects of this class on error.*
- class [std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >](#)
- class [std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >](#)
- struct [std::regex_traits< _Ch_type >](#)
Class regex_traits. Describes aspects of a regular expression.
- class [std::sub_match< _Biter >](#)

Typedefs

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`using std::__sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `typedef match_results< const char * > std::cmatch`
- `typedef regex_iterator< const char * > std::cregex_iterator`
- `typedef regex_token_iterator< const char * > std::cregex_token_iterator`

- typedef sub_match< const char * > [std::csub_match](#)
- typedef basic_regex< char > [std::regex](#)
- typedef match_results
 < string::const_iterator > **std::smatch**
- typedef regex_iterator
 < string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator
 < string::const_iterator > [std::sregex_token_iterator](#)
- typedef sub_match
 < string::const_iterator > [std::ssub_match](#)
- typedef match_results< const
 wchar_t * > **std::wcmatch**
- typedef regex_iterator< const
 wchar_t * > **std::wregex_iterator**
- typedef regex_token_iterator
 < const wchar_t * > [std::wregex_token_iterator](#)
- typedef sub_match< const
 wchar_t * > [std::wcsub_match](#)
- typedef basic_regex< wchar_t > [std::wregex](#)
- typedef match_results
 < wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator
 < wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator
 < wstring::const_iterator > [std::wsregex_token_iterator](#)
- typedef sub_match
 < wstring::const_iterator > [std::wssub_match](#)

Functions

- template<typename _Bi_iter >
 const sub_match< _Bi_iter > & **std::__unmatched_sub** ()
- bool [std::regex_traits< _Ch_type >::isctype](#) (_Ch_type __c, char_class_type __f) const
- template<typename _Biter >
 bool **std::operator!=** (const sub_match< _Biter > & __lhs, const sub_match< _Biter > & __rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
 bool **std::operator!=** (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
 bool **std::operator!=** (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs)
- template<typename _Bi_iter >
 bool **std::operator!=** (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs)
- template<typename _Bi_iter >
 bool **std::operator!=** (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)
- template<typename _Bi_iter >
 bool **std::operator!=** (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs)
- template<typename _Bi_iter >
 bool **std::operator!=** (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs)

- `template<typename _Bi_iter, class _Alloc >`
`bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc >`
`&__m2)`
- `template<typename _Bilter >`
`bool std::operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _`
`Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type,`
`_Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter >`
`&__m)`
- `template<typename _Bilter >`
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits,`
`_Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`
`_Bi_iter > &__rhs)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bilter >`
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`
- `int std::regex_traits< _Ch_type >::value (_Ch_type __ch, int __radix) const`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`

```

basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type,
_Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type >
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,
_Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex-
_constants::match_default)
• template<typename _Rx_traits , typename _Ch_type >
  basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _-
_Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __-
flags=regex_constants::match_default)

```

2.36.1 Detailed Description

A facility for performing regular expression pattern matching.

2.36.2 Typedef Documentation

2.36.2.1 `typedef regex_token_iterator<const char*> std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

Definition at line 2469 of file `regex.h`.

2.36.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 834 of file `regex.h`.

2.36.2.3 `typedef basic_regex<char> std::regex`

Standard regular expressions.

Definition at line 706 of file `regex.h`.

2.36.2.4 `typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator`

Token iterator for standard strings.

Definition at line 2472 of file `regex.h`.

2.36.2.5 `typedef sub_match<string::const_iterator> std::ssub_match`

Standard regex submatch over a standard string.

Definition at line 837 of file `regex.h`.

2.36.2.6 `typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator`

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2476 of file `regex.h`.

2.36.2.7 `typedef sub_match<const wchar_t*> std::wcsb_match`

Regex submatch over a C-style null-terminated wide string.

Definition at line 841 of file `regex.h`.

2.36.2.8 `typedef basic_regex<wchar_t> std::wregex`

Standard wide-character regular expressions.

Definition at line 710 of file regex.h.

2.36.2.9 `typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator`

Token iterator for standard wide-character strings.

Definition at line 2479 of file regex.h.

2.36.2.10 `typedef sub_match<wstring::const_iterator> std::wssub_match`

Regex submatch over a standard wide string.

Definition at line 844 of file regex.h.

2.36.3 Function Documentation**2.36.3.1** `template<typename _Ch_type> bool std::regex_traits<_Ch_type>::isctype (_Ch_type __c, char_class_type __f) const`

Determines if `c` is a member of an identified class.

Parameters

<code>__c</code>	a character.
<code>__f</code>	a class type (as returned from <code>lookup_classname</code>).

Returns

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

Exceptions

<code>std::bad_cast</code>	if the current locale does not have a ctype facet.
----------------------------	--

Definition at line 280 of file regex.h.

2.36.3.2 `template<typename _Bilter> bool std::operator!= (const sub_match<_Bilter> & __lhs, const sub_match<_Bilter> & __rhs) [inline]`

Tests the inequivalence of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 868 of file regex.h.

References `std::sub_match<_Bilter>::compare()`.

2.36.3.3 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator!=(const
__sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __lhs, const sub_match< _Bi_iter> & __rhs) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 943 of file `regex.h`.

2.36.3.4 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator!=(const sub_match<
_Bi_iter> & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1017 of file `regex.h`.

2.36.3.5 `template<typename _Bi_iter> bool std::operator!=(typename iterator_traits< _Bi_iter>::value_type const * __lhs, const
sub_match< _Bi_iter> & __rhs) [inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1091 of file `regex.h`.

2.36.3.6 `template<typename _Bi_iter> bool std::operator!=(const sub_match< _Bi_iter> & __lhs, typename iterator_traits<
_Bi_iter>::value_type const * __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1165 of file `regex.h`.

```
2.36.3.7 template<typename _Bi_iter > bool std::operator!= ( typename iterator_traits< _Bi_iter >::value_type const & __lhs, const
sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1242 of file `regex.h`.

```
2.36.3.8 template<typename _Bi_iter > bool std::operator!= ( const sub_match< _Bi_iter > & __lhs, typename iterator_traits<
_Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1322 of file `regex.h`.

```
2.36.3.9 template<typename _Bi_iter, class _Alloc > bool std::operator!= ( const match_results< _Bi_iter, _Alloc > & __m1, const
match_results< _Bi_iter, _Alloc > & __m2 ) [inline]
```

Compares two `match_results` for inequality.

Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1834 of file `regex.h`.

2.36.3.10 `template<typename _Bilter > bool std::operator< (const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 879 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`.

2.36.3.11 `template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 955 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`.

2.36.3.12 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator< (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1029 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

2.36.3.13 `template<typename _Bi_iter> bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1103 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.36.3.14 `template<typename _Bi_iter> bool std::operator< (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1177 of file `regex.h`.

2.36.3.15 `template<typename _Bi_iter> bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1254 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.36.3.16 `template<typename _Bi_iter> bool std::operator< (const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1334 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.36.3.17 `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter> basic_ostream<_Ch_type, _Ch_traits>& std::operator<< (basic_ostream<_Ch_type, _Ch_traits> & __os, const sub_match<_Bi_iter> & __m) [inline]`

Inserts a matched string into an output stream.

Parameters

<code>__os</code>	The output stream.
<code>__m</code>	A submatch string.

Returns

the output stream with the submatch string inserted.

Definition at line 1388 of file `regex.h`.

References `std::sub_match<_Bilter>::str()`.

2.36.3.18 `template<typename _Bilter> bool std::operator<= (const sub_match<_Bilter> & __lhs, const sub_match<_Bilter> & __rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 890 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.36.3.19 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator<= (const
__sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __lhs, const sub_match< _Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 991 of file `regex.h`.

2.36.3.20 `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator<= (const sub_match< _Bi_iter> &
__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1065 of file `regex.h`.

2.36.3.21 `template<typename _Bi_iter> bool std::operator<= (typename iterator_traits< _Bi_iter>::value_type const * __lhs,
const sub_match< _Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1139 of file `regex.h`.

2.36.3.22 `template<typename _Bi_iter> bool std::operator<= (const sub_match< _Bi_iter> & __lhs, typename iterator_traits<
_Bi_iter>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1213 of file `regex.h`.

2.36.3.23 `template<typename _Bi_iter> bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1293 of file `regex.h`.

2.36.3.24 `template<typename _Bi_iter> bool std::operator<= (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1373 of file `regex.h`.

2.36.3.25 `template<typename _Bilter> bool std::operator== (const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs) [inline]`

Tests the equivalence of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 857 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.36.3.26 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator==(const
__sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __lhs, const sub_match< _Bi_iter> & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 930 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, and `std::sub_match<_Biter>::compare()`.

2.36.3.27 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator==(const sub_match<
_Bi_iter> & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1004 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, and `std::sub_match<_Biter>::compare()`.

2.36.3.28 `template<typename _Bi_iter> bool std::operator==(typename iterator_traits< _Bi_iter>::value_type const * __lhs,
const sub_match< _Bi_iter> & __rhs) [inline]`

Tests the equivalence of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A C string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1078 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.36.3.29 `template<typename _Bi_iter> bool std::operator==(const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const * __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string?

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1152 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.36.3.30 `template<typename _Bi_iter> bool std::operator==(typename iterator_traits<_Bi_iter>::value_type const & __lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1226 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.36.3.31 `template<typename _Bi_iter> bool std::operator==(const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1306 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.36.3.32 `template<typename _Bi_iter, typename _Alloc> bool std::operator==(const match_results<_Bi_iter, _Alloc> & __m1, const match_results<_Bi_iter, _Alloc> & __m2) [inline]`

Compares two `match_results` for equality.

Returns

true if the two objects refer to the same match, false otherwise.

Definition at line 1810 of file `regex.h`.

References `std::match_results<_Bi_iter, _Alloc>::begin()`, `std::match_results<_Bi_iter, _Alloc>::empty()`, `std::match_results<_Bi_iter, _Alloc>::end()`, `std::equal()`, `std::match_results<_Bi_iter, _Alloc>::prefix()`, `std::match_results<_Bi_iter, _Alloc>::ready()`, `std::match_results<_Bi_iter, _Alloc>::size()`, and `std::match_results<_Bi_iter, _Alloc>::suffix()`.

2.36.3.33 `template<typename _Bilter> bool std::operator>(const sub_match<_Bilter> & __lhs, const sub_match<_Bilter> & __rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 912 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.36.3.34 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator>(const sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc> & __lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 967 of file `regex.h`.

2.36.3.35 `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator> (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1041 of file regex.h.

2.36.3.36 `template<typename _Bi_iter> bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1115 of file regex.h.

2.36.3.37 `template<typename _Bi_iter> bool std::operator> (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1189 of file regex.h.

2.36.3.38 `template<typename _Bi_iter> bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1269 of file `regex.h`.

2.36.3.39 `template<typename _Bi_iter > bool std::operator> (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1349 of file `regex.h`.

2.36.3.40 `template<typename _Biter > bool std::operator>= (const sub_match< _Biter > & __lhs, const sub_match< _Biter > & __rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 901 of file `regex.h`.

References `std::sub_match< _Biter >::compare()`.

2.36.3.41 `template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 979 of file `regex.h`.

```
2.36.3.42 template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator>= ( const sub_match< _Bi_iter> &
    __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1053 of file `regex.h`.

```
2.36.3.43 template<typename _Bi_iter> bool std::operator>= ( typename iterator_traits< _Bi_iter>::value_type const * __lhs,
    const sub_match< _Bi_iter> & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1127 of file `regex.h`.

```
2.36.3.44 template<typename _Bi_iter> bool std::operator>= ( const sub_match< _Bi_iter> & __lhs, typename iterator_traits<
    _Bi_iter>::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1201 of file `regex.h`.

2.36.3.45 `template<typename _Bi_iter> bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1281 of file `regex.h`.

2.36.3.46 `template<typename _Bi_iter> bool std::operator>= (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1361 of file `regex.h`.

2.36.3.47 `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags = regex_constants::match_default)`

Determines if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

Parameters

<code>__s</code>	Start of the character sequence to match.
<code>__e</code>	One-past-the-end of the character sequence to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Todo Implement this function.

Definition at line 1878 of file `regex.h`.

Referenced by `std::regex_match()`.

```
2.36.3.48 template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex_match ( _Bi_iter __first,
    _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __flags =
    regex_constants::match_default )
```

Indicates if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

Parameters

<i>__first</i>	Beginning of the character sequence to match.
<i>__last</i>	One-past-the-end of the character sequence to match.
<i>__re</i>	The regular expression.
<i>__flags</i>	Controls how the regular expression is matched.

Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 1909 of file `regex.h`.

References `std::regex_match()`.

```
2.36.3.49 template<typename _Ch_type, typename _Alloc, typename _Rx_traits> bool std::regex_match ( const _Ch_type
    * __s, match_results< const _Ch_type *, _Alloc> & __m, const basic_regex< _Ch_type, _Rx_traits> & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

<i>__s</i>	The C-style null-terminated string to match.
<i>__m</i>	The match results.
<i>__re</i>	The regular expression.
<i>__f</i>	Controls how the regular expression is matched.

Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 1934 of file `regex.h`.

References `std::regex_match()`.

```
2.36.3.50 template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool
std::regex_match ( const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string<
_Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re,
regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression `e` and a string.

Parameters

<code>__s</code>	The string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 1958 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```
2.36.3.51 template<typename _Ch_type, class _Rx_traits> bool std::regex_match ( const _Ch_type * __s, const basic_regex<
_Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f = regex_constants::match_default )
[inline]
```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 1981 of file `regex.h`.

References `std::regex_match()`.

```
2.36.3.52 template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits > bool
std::regex_match ( const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type,
_Rx_traits > & __re, regex_constants::match_flag_type __flags = regex_constants::match_default )
[inline]
```

Indicates if there is a match between the regular expression `e` and a string.

Parameters

<code>__s</code>	[IN] The string to match.
<code>__re</code>	[IN] The regular expression.
<code>__flags</code>	[IN] Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2003 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```
2.36.3.53 template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type > _Out_iter std::regex_replace
( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string<
_Ch_type > & __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default )
[inline]
```

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Parameters

<code>__out</code>	
<code>__first</code>	
<code>__last</code>	
<code>__e</code>	
<code>__fmt</code>	
<code>__flags</code>	

Returns

out

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Todo Implement this function.

Definition at line 2162 of file `regex.h`.

Referenced by `std::regex_replace()`.

2.36.3.54 `template<typename _Rx_traits, typename _Ch_type> basic_string<_Ch_type> std::regex_replace (const basic_string<_Ch_type> & __s, const basic_regex<_Ch_type, _Rx_traits> & __e, const basic_string<_Ch_type> & __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Parameters

<code>__s</code>	
<code>__e</code>	
<code>__fmt</code>	
<code>__flags</code>	

Returns

a copy of string `s` with replacements.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2182 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_replace()`.

2.36.3.55 `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results<_Bi_iter, _Alloc> & __m, const basic_regex<_Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Searches for a regular expression within a range.

Parameters

<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__m</code>	[OUT] The match results.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <i>m</i> is undefined.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Todo Implement this function.

Definition at line 2028 of file `regex.h`.

Referenced by `std::regex_search()`.

```
2.36.3.56 template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits > bool std::regex_search ( _Bi_iter __first,
    _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags =
    regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

Parameters

<i>__first</i>	[IN] The start of the string to search.
<i>__last</i>	[IN] One-past-the-end of the string to search.
<i>__re</i>	[IN] The regular expression to search for.
<i>__flags</i>	[IN] Search policy flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2049 of file `regex.h`.

References `std::regex_search()`.

```
2.36.3.57 template<typename _Ch_type, class _Alloc, class _Rx_traits > bool std::regex_search ( const _Ch_type *
    __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

<i>__s</i>	[IN] A C-string to search for the regex.
<i>__m</i>	[OUT] The set of regex matches.
<i>__e</i>	[IN] The regex to search for in <i>s</i> .
<i>__f</i>	[IN] The search flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of m is undefined.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2073 of file `regex.h`.

References `std::regex_search()`.

2.36.3.58 `template<typename _Ch_type, typename _Rx_traits> bool std::regex_search (const _Ch_type * __s, const basic_regex<_Ch_type, _Rx_traits> & __e, regex_constants::match_flag_type __f = regex_constants::match_default)`
`[inline]`

Searches for a regular expression within a C-string.

Parameters

<code>__s</code>	[IN] The C-string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__f</code>	[IN] Search policy flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2093 of file `regex.h`.

References `std::regex_search()`.

2.36.3.59 `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits> bool std::regex_search (const basic_string<_Ch_type, _Ch_traits, _String_allocator> & __s, const basic_regex<_Ch_type, _Rx_traits> & __e, regex_constants::match_flag_type __flags = regex_constants::match_default)`
`[inline]`

Searches for a regular expression within a string.

Parameters

<code>__s</code>	[IN] The string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2113 of file `regex.h`.

References `std::regex_search()`.

```
2.36.3.60  template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >
            bool std::regex_search ( const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename
            basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits
            > & __e, regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

Parameters

<i>__s</i>	[IN] A C++ string to search for the regex.
<i>__m</i>	[OUT] The set of regex matches.
<i>__e</i>	[IN] The regex to search for in <i>s</i> .
<i>__f</i>	[IN] The search flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <i>m</i> is undefined.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2136 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

```
2.36.3.61  template<typename _Ch_type, typename _Rx_traits > void std::swap ( basic_regex< _Ch_type, _Rx_traits > & __lhs,
            basic_regex< _Ch_type, _Rx_traits > & __rhs ) [inline]
```

Swaps the contents of two regular expression objects.

Parameters

<i>__lhs</i>	First regular expression.
<i>__rhs</i>	Second regular expression.

Definition at line 722 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::swap()`.

2.36.3.62 `template<typename _Bi_iter, typename _Alloc> void std::swap (match_results<_Bi_iter, _Alloc> & __lhs, match_results<_Bi_iter, _Alloc> & __rhs) [inline]`

Swaps two match results.

Parameters

<code>__lhs</code>	A match result.
<code>__rhs</code>	A match result.

The contents of the two `match_results` objects are swapped.

Definition at line 1848 of file `regex.h`.

References `std::match_results<_Bi_iter, _Alloc>::swap()`.

2.36.3.63 `template<typename _Ch_type> int std::regex_traits<_Ch_type>::value (_Ch_type __ch, int __radix) const`

Converts a digit to an int.

Parameters

<code>__ch</code>	a character representing a digit.
<code>__radix</code>	the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

Definition at line 314 of file `regex.h`.

References `std::basic_ios<_CharT, _Traits>::fail()`, `std::hex()`, and `std::oct()`.

2.37 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- class `std::__detail::_Automaton`
Base class for, um, automata. Could be an NFA or a DFA. Your choice.
- struct `std::__detail::_CharMatcher<_InIterT, _TraitsT>`
Matches a single character.
- class `std::__detail::_Compiler<_InIter, _TraitsT>`
Builds an NFA from an input iterator interval.
- struct `std::__detail::_EndTagger<_FwdIterT, _TraitsT>`
End state tag.
- class `std::__detail::_Grep_matcher`
Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.
- class `std::__detail::_Nfa`
struct _Nfa
- struct `std::__detail::_PatternCursor`
ABC for pattern matching.
- struct `std::__detail::_RangeMatcher<_InIterT, _TraitsT>`
Matches a character range (bracket expression)
- struct `std::__detail::_Results`
Provides a generic facade for a templated match_results.
- class `std::__detail::_Scanner<_InputIterator>`
struct _Scanner. Scans an input range for regex tokens.
- struct `std::__detail::_Scanner_base`
Base class for scanner.
- class `std::__detail::_SpecializedCursor<_FwdIterT>`
Provides a cursor into the specific target string.
- class `std::__detail::_SpecializedResults<_FwdIterT, _Alloc>`
A _Results facade specialized for wrapping a templated match_results.
- struct `std::__detail::_StartTagger<_FwdIterT, _TraitsT>`
Start state tag.
- struct `std::__detail::_State`
struct _State
- class `std::__detail::_StateSeq`
Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Typedefs

- typedef `std::shared_ptr`
`<_Automaton> std::__detail::_AutomatonPtr`
- typedef `std::function< bool(const`
`_PatternCursor &)> std::__detail::_Matcher`
- typedef `int std::__detail::_StateIdT`
- typedef `std::set<_StateIdT> std::__detail::_StateSet`
- typedef `std::stack<_StateIdT,`
`std::vector<_StateIdT>> std::__detail::_StateStack`
- typedef `std::function< void(const`
`_PatternCursor &, _Results &)> std::__detail::_Tagger`

Enumerations

- enum `std::__detail::_Opcode` {
`_S_opcode_unknown, _S_opcode_alternative, _S_opcode_subexpr_begin, _S_opcode_subexpr_end,`
`_S_opcode_match, _S_opcode_accept }`

Functions

- `std::__detail::_Compiler<_InIter, _TraitsT>::_Compiler` (const `_InIter` &__b, const `_InIter` &__e, `_TraitsT` &__traits, `_FlagT` __flags)
- `std::__detail::_SpecializedResults<_FwdIterT, _Alloc>::_SpecializedResults` (const `_Automaton::SizeT` __size, const `_SpecializedCursor<_FwdIterT>` &__cursor, `match_results<_FwdIterT, _Alloc>` &__m)
- template<typename `_InIter`, typename `_TraitsT`>
`_AutomatonPtr std::__detail::_compile` (const `_InIter` &__b, const `_InIter` &__e, `_TraitsT` &__t, `regex_constants::syntax_option_type` __f)
- template<typename `_FwdIterT`>
`_SpecializedCursor<_FwdIterT> std::__detail::_cursor` (const `_FwdIterT` &__b, const `_FwdIterT` __e)
- bool `std::__detail::_AnyMatcher` (const `_PatternCursor` &)
- void `std::__detail::_Scanner<_InputIterator>::_M_advance` ()
- void `std::__detail::_SpecializedResults<_FwdIterT, _Alloc>::_M_set_pos` (int __i, int __j, const `_PatternCursor` &__pc)

Variables

- static const `_StateIdT std::__detail::_S_invalid_state_id`

2.37.1 Detailed Description

2.37.2 Typedef Documentation

2.37.2.1 typedef `std::shared_ptr<_Automaton> std::__detail::_AutomatonPtr`

Generic shared pointer to an automaton.

Definition at line 62 of file `regex_nfa.h`.

2.37.2.2 `typedef std::function<bool (const _PatternCursor&)> std::__detail::_Matcher`

Indicates if current state matches cursor current.

Definition at line 120 of file `regex_nfa.h`.

2.37.2.3 `typedef int std::__detail::_StateldT`

Identifies a state in the NFA.

Definition at line 195 of file `regex_nfa.h`.

2.37.2.4 `typedef std::set<_StateldT> std::__detail::_StateSet`

The Grep Matcher works on sets of states. Here are sets of states.

Definition at line 251 of file `regex_nfa.h`.

2.37.2.5 `typedef std::stack<_StateldT, std::vector<_StateldT> > std::__detail::_StateStack`

A stack of states used in evaluating the NFA.

Definition at line 105 of file `regex_grep_matcher.h`.

2.37.2.6 `typedef std::function<void (const _PatternCursor&, _Results&)> std::__detail::_Tagger`

Tags current state (for subexpr begin/end).

Definition at line 84 of file `regex_nfa.h`.

2.37.3 Enumeration Type Documentation**2.37.3.1** `enum std::__detail::_Opcode`

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

Definition at line 66 of file `regex_nfa.h`.

2.37.4 Function Documentation**2.37.4.1** `bool std::__detail::_AnyMatcher (const _PatternCursor &) [inline]`

Matches any character.

Definition at line 124 of file `regex_nfa.h`.

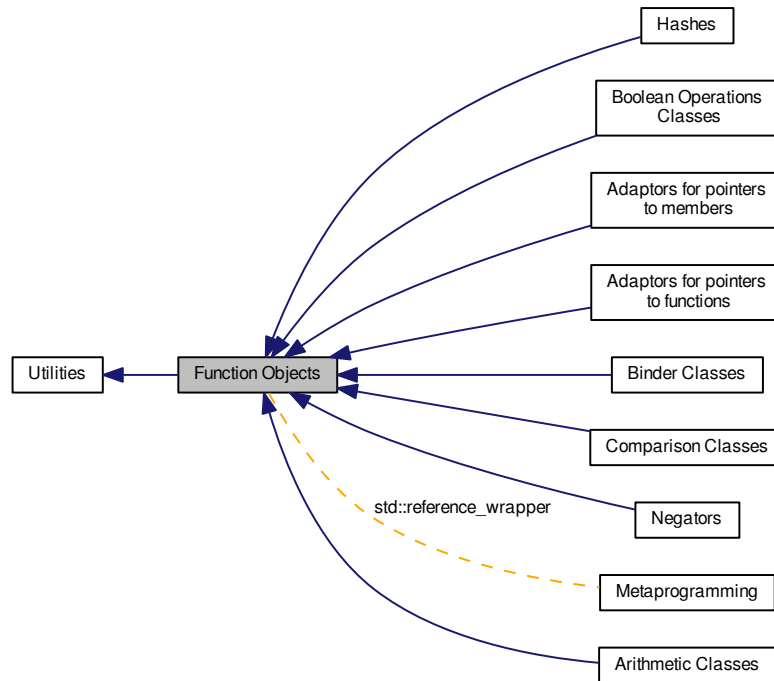
2.37.5 Variable Documentation**2.37.5.1** `const _StateldT std::__detail::_S_invalid_state_id [static]`

The special case in which a state identifier is not an index.

Definition at line 198 of file `regex_nfa.h`.

2.38 Function Objects

Collaboration diagram for Function Objects:



Modules

- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)
- [Arithmetic Classes](#)
- [Binder Classes](#)
- [Boolean Operations Classes](#)
- [Comparison Classes](#)
- [Hashes](#)
- [Negators](#)

Classes

- `struct std::binary_function<_Arg1, _Arg2, _Result>`
- `class std::function<_Res\(_ArgTypes...\)>`
*Primary class template for `std::function`.
 Polymorphic function wrapper.*
- `class std::reference_wrapper<_Tp>`
Primary class template for `reference_wrapper`.
- `struct std::unary_function<_Arg, _Result>`

Functions

- `template<typename _Tp, typename _Class >
_Mem_fn<_Tp _Class::* > std::mem_fn (_Tp _Class::* __pm) noexcept`

2.38.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

2.38.2 Function Documentation

2.38.2.1 `template<typename _Tp, typename _Class > _Mem_fn<_Tp _Class::* > std::mem_fn (_Tp _Class::* __pm)
[inline], [noexcept]`

Returns a function object that forwards to the member pointer `pm`.

Definition at line 961 of file `functional`.

2.39 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



Classes

- struct `std::divides<_Tp>`
One of the [math functors](#).
- struct `std::minus<_Tp>`
One of the [math functors](#).
- struct `std::modulus<_Tp>`
One of the [math functors](#).
- struct `std::multiplies<_Tp>`
One of the [math functors](#).
- struct `std::negate<_Tp>`
One of the [math functors](#).
- struct `std::plus<_Tp>`
One of the [math functors](#).

2.39.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

2.40 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct `std::equal_to<_Tp>`
One of the [comparison functors](#).
- struct `std::greater<_Tp>`
One of the [comparison functors](#).
- struct `std::greater_equal<_Tp>`
One of the [comparison functors](#).
- struct `std::less<_Tp>`
One of the [comparison functors](#).
- struct `std::less_equal<_Tp>`
One of the [comparison functors](#).
- struct `std::not_equal_to<_Tp>`
One of the [comparison functors](#).

2.40.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

2.41 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

- struct `std::logical_and<_Tp>`
One of the [Boolean operations functors](#).
- struct `std::logical_not<_Tp>`
One of the [Boolean operations functors](#).
- struct `std::logical_or<_Tp>`
One of the [Boolean operations functors](#).

2.41.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

2.42 Negators

Collaboration diagram for Negators:



Classes

- class `std::binary_negate<_Predicate>`
One of the *negation functors*.
- class `std::unary_negate<_Predicate>`
One of the *negation functors*.

Functions

- `template<typename _Predicate>`
`unary_negate<_Predicate> std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate>`
`binary_negate<_Predicate> std::not2 (const _Predicate &__pred)`

2.42.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

2.42.2 Function Documentation

2.42.2.1 `template<typename _Predicate> unary_negate<_Predicate> std::not1 (const _Predicate & __pred) [inline]`

One of the *negation functors*.

Definition at line 369 of file `stl_function.h`.

2.42.2.2 `template<typename _Predicate > binary_negate<_Predicate> std::not2 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 394 of file `stl_function.h`.

2.43 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`
One of the [adaptors for function pointers](#).
- class `std::pointer_to_unary_function<_Arg, _Result>`
One of the [adaptors for function pointers](#).

Functions

- `template<typename _Arg, typename _Result>`
`pointer_to_unary_function`
`<_Arg, _Result> std::ptr_fun (_Result(*)(__x))(_Arg)`
- `template<typename _Arg1, typename _Arg2, typename _Result>`
`pointer_to_binary_function`
`<_Arg1, _Arg2, _Result> std::ptr_fun (_Result(*)(__x)(_Arg1, _Arg2))`

2.43.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

2.43.2 Function Documentation

2.43.2.1 `template<typename _Arg, typename _Result> pointer_to_unary_function<_Arg, _Result> std::ptr_fun (_Result(*)(__x)) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 442 of file `stl_function.h`.

```
2.43.2.2  template<typename _Arg1 , typename _Arg2 , typename _Result > pointer_to_binary_function<_Arg1, _Arg2, _Result>  
          std::ptr_fun ( _Result(*)( _Arg1, _Arg2) __x )  [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 468 of file stl_function.h.

2.44 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`
One of the adaptors for member pointers.
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`
One of the adaptors for member pointers.
- class `std::const_mem_fun_ref_t<_Ret, _Tp>`
One of the adaptors for member pointers.
- class `std::const_mem_fun_t<_Ret, _Tp>`
One of the adaptors for member pointers.
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`
One of the adaptors for member pointers.
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`
One of the adaptors for member pointers.
- class `std::mem_fun_ref_t<_Ret, _Tp>`
One of the adaptors for member pointers.
- class `std::mem_fun_t<_Ret, _Tp>`
One of the adaptors for member pointers.

Functions

- `template<typename _Ret, typename _Tp>`
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp>`
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`

2.44.1 Detailed Description

There are a total of $8 = 2^3$ function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

2.45 Heap

Collaboration diagram for Heap:



Functions

- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

2.45.1 Detailed Description

2.45.2 Function Documentation

2.45.2.1 `template<typename _RandomAccessIterator > bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Determines whether a range is a heap.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

True if range is a heap, false otherwise.

Definition at line 565 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.45.2.2 `template<typename _RandomAccessIterator, typename _Compare> bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Determines whether a range is a heap using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

Returns

True if range is a heap, false otherwise.

Definition at line 578 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.45.2.3 `template<typename _RandomAccessIterator> _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Search the end of a heap.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is a heap.

Definition at line 517 of file `stl_heap.h`.

References `std::distance()`.

2.45.2.4 `template<typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Search the end of a heap using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [`__first`, `__last`) for which the range [`__first`, *i*) is a heap. Comparisons are made using `__comp`.

Definition at line 543 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

2.45.2.5 `template<typename _RandomAccessIterator > void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

Construct a heap over a range.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation makes the elements in [`__first`, `__last`) into a heap.

Definition at line 380 of file `stl_heap.h`.

2.45.2.6 `template<typename _RandomAccessIterator, typename _Compare > void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

Construct a heap over a range using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation makes the elements in [`__first`, `__last`) into a heap. Comparisons are made using `__comp`.

Definition at line 420 of file `stl_heap.h`.

Referenced by `std::__heap_select()`, `std::partial_sort_copy()`, and `std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue()`.

2.45.2.7 `template<typename _RandomAccessIterator > void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Pop an element off a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

Precondition

`[__first, __last)` is a valid, non-empty range.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap.

Definition at line 281 of file `stl_heap.h`.

2.45.2.8 `template<typename _RandomAccessIterator, typename _Compare> void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Pop an element off a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap. Comparisons are made using `comp`.

Definition at line 356 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`.

2.45.2.9 `template<typename _RandomAccessIterator> void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Push an element onto a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.

This operation pushes the element at `last-1` onto the valid heap over the range `[__first, __last-1)`. After completion, `[__first, __last)` is a valid heap.

Definition at line 156 of file `stl_heap.h`.

2.45.2.10 `template<typename _RandomAccessIterator, typename _Compare> void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Push an element onto a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.
<code>__comp</code>	Comparison functor.

This operation pushes the element at `__last-1` onto the valid heap over the range `[__first, __last-1)`. After completion, `[__first, __last)` is a valid heap. Compare operations are performed using `comp`.

Definition at line 206 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::push()`.

2.45.2.11 `template<typename _RandomAccessIterator > void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

Sort a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation sorts the valid heap in the range `[__first,__last)`.

Definition at line 459 of file `stl_heap.h`.

2.45.2.12 `template<typename _RandomAccessIterator, typename _Compare > void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

Sort a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

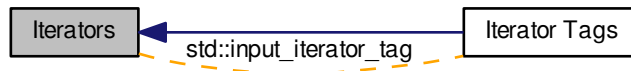
This operation sorts the valid heap in the range `[__first,__last)`. Comparisons are made using `__comp`.

Definition at line 488 of file `stl_heap.h`.

Referenced by `std::partial_sort()`, and `std::partial_sort_copy()`.

2.46 Iterators

Collaboration diagram for Iterators:



Modules

- [Iterator Tags](#)

Classes

- class `std::__has_iterator_category_helper< _Tp >`
Traits class for iterators.
- class `std::back_insert_iterator< _Container >`
Turns assignment into insertion.
- class `std::front_insert_iterator< _Container >`
Turns assignment into insertion.
- struct `std::input_iterator_tag`
Marking input iterators.
- class `std::insert_iterator< _Container >`
Turns assignment into insertion.
- class `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`
Provides input iterator semantics for streams.
- class `std::istreambuf_iterator< _CharT, _Traits >`
Provides input iterator semantics for streambufs.
- struct `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
Common iterator class.
- struct `std::iterator_traits< _Tp * >`
Partial specialization for pointer types.
- struct `std::iterator_traits< const _Tp * >`
Partial specialization for const pointer types.
- class `std::move_iterator< _Iterator >`
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`
Provides output iterator semantics for streams.
- class `std::ostreambuf_iterator< _CharT, _Traits >`
Provides output iterator semantics for streambufs.
- class `std::reverse_iterator< _Iterator >`

Functions

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if`
`< __is_char< _CharT >::__value,`
`ostreambuf_iterator< _CharT >`
`>::__type std::copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if`
`< __is_char< _CharT >::__value,`
`ostreambuf_iterator< _CharT >`
`>::__type std::copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT`
`> __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if`
`< __is_char< _CharT >::__value,`
`_CharT * >::__type std::copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT`
`> __last, _CharT * __result)`
- `template<typename _Iter >`
`iterator_traits< _Iter >`
`::iterator_category std::iterator_category (const _Iter &)`
- `template<typename _Iterator , typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_`
`Iterator>::__value_type>::__value, _Iterator, move_iterator<_Iterator>::__type>`
`_ReturnType std::make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Container >`
`back_insert_iterator< _Container > std::back_inserter (_Container & __x)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if`
`< __is_char< _CharT >::__value,`
`ostreambuf_iterator< _CharT >`
`>::__type std::copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf-`
`_iterator< _CharT > __result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if`
`< __is_char< _CharT >::__value,`
`istreambuf_iterator< _CharT >`
`>::__type std::find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT`
`& __val)`
- `template<typename _Container >`
`front_insert_iterator< _Container > std::front_inserter (_Container & __x)`
- `template<typename _Container , typename _Iterator >`
`insert_iterator< _Container > std::inserter (_Container & __x, _Iterator __i)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::make_move_iterator (_Iterator __i)`
- `template<class _Tp , class _CharT , class _Traits , class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp,`
`_CharT, _Traits, _Dist > & __y)`
- `template<typename _CharT , typename _Traits >`
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT,`
`_Traits > & __b)`
- `template<typename _Iterator >`
`bool std::operator!= (const reverse_iterator< _Iterator > & __x, const reverse_iterator< _Iterator > & __y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool std::operator!= (const reverse_iterator< _IteratorL > & __x, const reverse_iterator< _IteratorR > & __y)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n,`
`const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const`
`move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >`
`::difference_type std::operator- (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) ->`
`decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) ->`
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`auto std::operator- (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) ->`
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp,`
`_CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`
`_Traits > &__b)`
- `template<typename _Iterator >`
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

- `template<typename _Iterator >`
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

2.46.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

2.46.2 Function Documentation

2.46.2.1 `template<typename _Iter > iterator_traits<_Iter>::iterator_category std::__iterator_category (const _Iter &)`
`[inline]`

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 201 of file `stl_iterator_base_types.h`.

Referenced by `std::__find_if_not()`, `__gnu_debug::__valid_range_aux()`, `std::advance()`, `__gnu_cxx::copy_n()`, `std::copy_n()`, `__gnu_cxx::distance()`, `std::distance()`, `std::find()`, `std::find_end()`, `std::find_if()`, `std::partition()`, `std::reverse()`, `std::search_n()`, `__gnu_cxx::uninitialized_copy_n()`, `std::uninitialized_copy_n()`, and `std::unique_copy()`.

2.46.2.2 `template<typename _Container > back_insert_iterator<_Container> std::back_inserter (_Container & __x)`
`[inline]`

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `back_insert_iterator` working on `__x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 479 of file `stl_iterator.h`.

Referenced by `std::match_results< _FwdIterT, _Alloc >::format()`, and `std::regex_replace()`.

2.46.2.3 `template<typename _Container > front_insert_iterator<_Container> std::front_inserter (_Container & __x)`
`[inline]`

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `front_insert_iterator` working on `x`.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 569 of file `stl_iterator.h`.

```
2.46.2.4 template<typename _Container, typename _Iterator> insert_iterator<_Container> std::inserter ( _Container & __x,
    _Iterator __i ) [inline]
```

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `insert_iterator` working on `__x`.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 683 of file `stl_iterator.h`.

```
2.46.2.5 template<class _Tp, class _CharT, class _Traits, class _Dist> bool std::operator!= ( const istream_iterator< _Tp, _CharT,
    _Traits, _Dist> & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist> & __y ) [inline]
```

Return false if `x` and `y` are both end or not end, or `x` and `y` are the same.

Definition at line 137 of file `stream_iterator.h`.

```
2.46.2.6 template<typename _Tp, typename _CharT, typename _Traits, typename _Dist> bool std::operator== ( const
    istream_iterator< _Tp, _CharT, _Traits, _Dist> & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist> & __y )
    [inline]
```

Return true if `x` and `y` are both end or not end, or `x` and `y` are the same.

Definition at line 130 of file `stream_iterator.h`.

```
2.46.2.7 template<typename _Iterator> bool std::operator== ( const reverse_iterator< _Iterator> & __x, const reverse_iterator<
    _Iterator> & __y ) [inline]
```

Parameters

<code>__x</code>	A <code>reverse_iterator</code> .
<code>__y</code>	A <code>reverse_iterator</code> .

Returns

A simple bool.

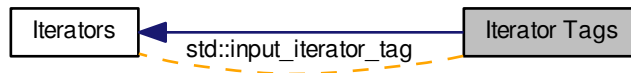
Reverse iterators forward many operations to their underlying `base()` iterators. Others are implemented in terms of one another.

Definition at line 291 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::base()`.

2.47 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

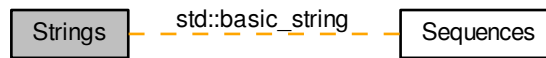
- struct `std::bidirectional_iterator_tag`
Bidirectional iterators support a superset of forward iterator operations.
- struct `std::forward_iterator_tag`
Forward iterators support a superset of input iterator operations.
- struct `std::input_iterator_tag`
Marking input iterators.
- struct `std::output_iterator_tag`
Marking output iterators.
- struct `std::random_access_iterator_tag`
Random-access iterators support a superset of bidirectional iterator operations.

2.47.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

2.48 Strings

Collaboration diagram for Strings:



Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
Managing sequences of characters and character-like objects.
- struct `std::char_traits<_CharT>`
Basis for explicit traits specializations.

Typedefs

- typedef `basic_string<char>` `std::string`
- typedef `basic_string<char16_t>` `std::u16string`
- typedef `basic_string<char32_t>` `std::u32string`
- typedef `basic_string<wchar_t>` `std::wstring`

2.48.1 Detailed Description

2.48.2 Typedef Documentation

2.48.2.1 typedef `basic_string<char>` `std::string`

A string of `char`.

Definition at line 64 of file `stringfwd.h`.

2.48.2.2 typedef `basic_string<char16_t>` `std::u16string`

A string of `char16_t`.

Definition at line 80 of file `stringfwd.h`.

2.48.2.3 typedef `basic_string<char32_t>` `std::u32string`

A string of `char32_t`.

Definition at line 83 of file `stringfwd.h`.

2.48.2.4 typedef `basic_string<wchar_t>` `std::wstring`

A string of `wchar_t`.

Definition at line 70 of file `stringfwd.h`.

2.49 Binder Classes

Collaboration diagram for Binder Classes:



Namespaces

- namespace `std::placeholders`

Classes

- struct `std::_Placeholder<_Num>`
The type of placeholder objects defined by `libstdc++`.
- class `std::binder1st<_Operation>`
One of the *binder functors*.
- class `std::binder2nd<_Operation>`
One of the *binder functors*.
- struct `std::is_bind_expression<_Tp>`
Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].
- struct `std::is_bind_expression<_Bind<_Signature>>>`
Class template `_Bind` is always a bind expression.
- struct `std::is_bind_expression<_Bind_result<_Result,_Signature>>>`
Class template `_Bind_result` is always a bind expression.
- struct `std::is_bind_expression<const _Bind<_Signature>>>`
Class template `_Bind` is always a bind expression.
- struct `std::is_bind_expression<const _Bind_result<_Result,_Signature>>>`
Class template `_Bind_result` is always a bind expression.
- struct `std::is_bind_expression<const volatile _Bind<_Signature>>>`
Class template `_Bind` is always a bind expression.
- struct `std::is_bind_expression<const volatile _Bind_result<_Result,_Signature>>>`
Class template `_Bind_result` is always a bind expression.
- struct `std::is_bind_expression<volatile _Bind<_Signature>>>`
Class template `_Bind` is always a bind expression.
- struct `std::is_bind_expression<volatile _Bind_result<_Result,_Signature>>>`
Class template `_Bind_result` is always a bind expression.
- struct `std::is_placeholder<_Tp>`
Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct `std::is_placeholder<_Placeholder<_Num>>>`

Functions

- `template<typename _Func, typename... _BoundArgs>
_Bind_helper<__is_socketlike
<_Func>::value, _Func,
_BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>
_Bindres_helper<_Result,
_Func, _BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Operation, typename _Tp >
binder1st<_Operation > std::binder1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp >
binder2nd<_Operation > std::binder2nd (const _Operation &__fn, const _Tp &__x)`

2.49.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B's operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `binder1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>(), 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `binder1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `binder1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `binder1st(std::plus<int>(), 5)`).

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `binder1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `binder2nd` by `std::bind(f, std::placeholders::_1, x)`.

2.49.2 Function Documentation

2.49.2.1 `template<typename _Func, typename... _BoundArgs> _Bind_helper<__is_socketlike<_Func>::value, _Func,
_BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&... __args) [inline]`

Function template for `std::bind`.

Definition at line 1655 of file `functional`.

Referenced by `std::is_permutation()`.

2.49.2.2 `template<typename _Result, typename _Func, typename... _BoundArgs> _Bindres_helper<_Result, _Func,
_BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&... __args) [inline]`

Function template for `std::bind<R>`.

Definition at line 1682 of file `functional`.

```
2.49.2.3  template<typename _Operation , typename _Tp > binder1st<_Operation> std::bind1st ( const _Operation & __fn, const  
         _Tp & __x ) [inline]
```

One of the [binder functors](#).

Definition at line 131 of file binders.h.

```
2.49.2.4  template<typename _Operation , typename _Tp > binder2nd<_Operation> std::bind2nd ( const _Operation & __fn, const  
         _Tp & __x ) [inline]
```

One of the [binder functors](#).

Definition at line 166 of file binders.h.

2.50 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int __n, unsigned int __m, _Tp __x)
- `float std::tr1::assoc_laguerref` (unsigned int __n, unsigned int __m, float __x)
- `long double std::tr1::assoc_laguerrel` (unsigned int __n, unsigned int __m, long double __x)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int __l, unsigned int __m, _Tp __x)
- `float std::tr1::assoc_legendref` (unsigned int __l, unsigned int __m, float __x)
- `long double std::tr1::assoc_legendrel` (unsigned int __l, unsigned int __m, long double __x)
- `template<typename _Tpx, typename _Tpy >`
`__gnu_cxx::__promote_2< _Tpx,`
`_Tpy >::__type std::tr1::beta` (_Tpx __x, _Tpy __y)
- `float std::tr1::betaf` (float __x, float __y)
- `long double std::tr1::betal` (long double __x, long double __y)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (_Tp __k)
- `float std::tr1::comp_ellint_1f` (float __k)
- `long double std::tr1::comp_ellint_1l` (long double __k)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (_Tp __k)
- `float std::tr1::comp_ellint_2f` (float __k)
- `long double std::tr1::comp_ellint_2l` (long double __k)
- `template<typename _Tp, typename _Tpn >`
`__gnu_cxx::__promote_2< _Tp,`
`_Tpn >::__type std::tr1::comp_ellint_3` (_Tp __k, _Tpn __nu)
- `float std::tr1::comp_ellint_3f` (float __k, float __nu)
- `long double std::tr1::comp_ellint_3l` (long double __k, long double __nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa,`
`_Tpc, _Tp >::__type std::tr1::conf_hyperg` (_Tpa __a, _Tpc __c, _Tp __x)
- `float std::tr1::conf_hypergf` (float __a, float __c, float __x)
- `long double std::tr1::conf_hypergl` (long double __a, long double __c, long double __x)
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu,`
`_Tp >::__type std::tr1::cyl_bessel_i` (_Tpnu __nu, _Tp __x)
- `float std::tr1::cyl_bessel_if` (float __nu, float __x)

- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu,
_Tp >::__type **std::tr1::cyl_bessel_j** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu,
_Tp >::__type **std::tr1::cyl_bessel_k** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu,
_Tp >::__type **std::tr1::cyl_neumann** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp >
__gnu_cxx::__promote_2< _Tp,
_Tpp >::__type **std::tr1::ellint_1** (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp >
__gnu_cxx::__promote_2< _Tp,
_Tpp >::__type **std::tr1::ellint_2** (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp >
__gnu_cxx::__promote_3< _Tp,
_Tpn, _Tpp >::__type **std::tr1::ellint_3** (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::expint** (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::hermite** (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_4< _Tpa,
_Tpb, _Tpc, _Tp >::__type **std::tr1::hyperg** (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **std::tr1::hypergff** (float __a, float __b, float __c, float __x)
- long double **std::tr1::hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::laguerre** (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::legendre** (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)

- `template<typename _Tp >`
`__gnu_cxx::__promote<_Tp>::__type std::tr1::riemann_zeta (_Tp __x)`
- `float std::tr1::riemann_zetaf (float __x)`
- `long double std::tr1::riemann_zetal (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)`
- `float std::tr1::sph_besself (unsigned int __n, float __x)`
- `long double std::tr1::sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float std::tr1::sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double std::tr1::sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`
`__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)`
- `float std::tr1::sph_neumannf (unsigned int __n, float __x)`
- `long double std::tr1::sph_neumannl (unsigned int __n, long double __x)`

2.50.1 Detailed Description

A collection of advanced mathematical special functions.

2.50.2 Function Documentation

2.50.2.1 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x) [inline]`

5.2.1.1 Associated Laguerre polynomials.

Definition at line 1055 of file tr1/cmath.

2.50.2.2 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x) [inline]`

5.2.1.2 Associated Legendre functions.

Definition at line 1072 of file tr1/cmath.

2.50.2.3 `template<typename _Tpx, typename _Tpy > __gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta (_Tpx __x, _Tpy __y) [inline]`

5.2.1.3 Beta functions.

Definition at line 1089 of file tr1/cmath.

2.50.2.4 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_1 (_Tp __k) [inline]`

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 1106 of file tr1/cmath.

2.50.2.5 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_2 (_Tp __k) [inline]`

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 1123 of file tr1/cmath.

2.50.2.6 `template<typename _Tp, typename _Tpn> __gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu) [inline]`

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 1140 of file tr1/cmath.

2.50.2.7 `template<typename _Tpa, typename _Tpc, typename _Tp> __gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type std::tr1::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x) [inline]`

5.2.1.7 Confluent hypergeometric functions.

Definition at line 1157 of file tr1/cmath.

2.50.2.8 `template<typename _Tpnu, typename _Tp> __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 1174 of file tr1/cmath.

2.50.2.9 `template<typename _Tpnu, typename _Tp> __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 1191 of file tr1/cmath.

2.50.2.10 `template<typename _Tpnu, typename _Tp> __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 1208 of file tr1/cmath.

2.50.2.11 `template<typename _Tpnu, typename _Tp> __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.11 Cylindrical Neumann functions.

Definition at line 1225 of file tr1/cmath.

2.50.2.12 `template<typename _Tp, typename _Tpp> __gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi) [inline]`

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 1242 of file tr1/cmath.

2.50.2.13 `template<typename _Tp, typename _Tpp> __gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi) [inline]`

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 1259 of file tr1/cmath.

2.50.2.14 `template<typename _Tp, typename _Tpn, typename _Tpp> __gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi) [inline]`

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 1276 of file tr1/cmath.

2.50.2.15 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::expint (_Tp __x) [inline]`

5.2.1.15 Exponential integrals.

Definition at line 1293 of file tr1/cmath.

2.50.2.16 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::hermite (unsigned int __n, _Tp __x) [inline]`

5.2.1.16 Hermite polynomials.

Definition at line 1310 of file tr1/cmath.

2.50.2.17 `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp> __gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x) [inline]`

5.2.1.17 Hypergeometric functions.

Definition at line 1327 of file tr1/cmath.

2.50.2.18 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::laguerre (unsigned int __n, _Tp __x) [inline]`

5.2.1.18 Laguerre polynomials.

Definition at line 1344 of file tr1/cmath.

2.50.2.19 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::legendre (unsigned int __n, _Tp __x) [inline]`

5.2.1.19 Legendre polynomials.

Definition at line 1361 of file tr1/cmath.

2.50.2.20 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::riemann_zeta (_Tp __x) [inline]`

5.2.1.20 Riemann zeta function.

Definition at line 1378 of file tr1/cmath.

2.50.2.21 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x) [inline]`

5.2.1.21 Spherical Bessel functions.

Definition at line 1395 of file tr1/cmath.

2.50.2.22 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta) [inline]`

5.2.1.22 Spherical associated Legendre functions.

Definition at line 1412 of file tr1/cmath.

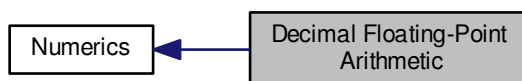
2.50.2.23 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x) [inline]`

5.2.1.23 Spherical Neumann functions.

Definition at line 1429 of file tr1/cmath.

2.51 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

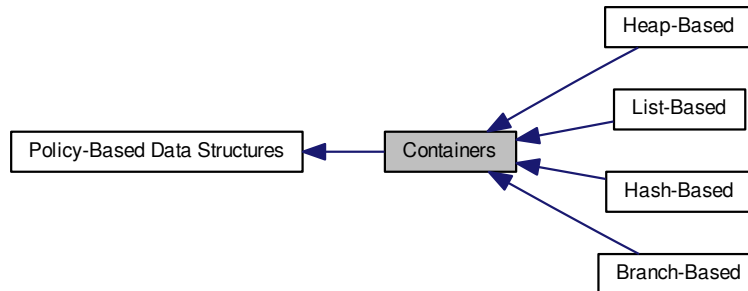
- namespace `std::decimal`

2.51.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

2.52 Containers

Collaboration diagram for Containers:



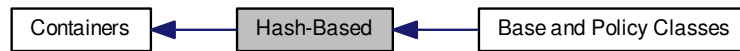
Modules

- [Branch-Based](#)
- [Hash-Based](#)
- [Heap-Based](#)
- [List-Based](#)

2.52.1 Detailed Description

2.53 Hash-Based

Collaboration diagram for Hash-Based:



Modules

- [Base and Policy Classes](#)

Classes

- `class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`
- `class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`
- `class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`

Macros

- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`

2.53.1 Detailed Description

2.54 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



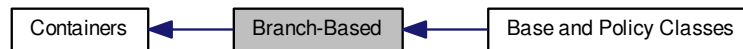
Classes

- class `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`
- class `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`

2.54.1 Detailed Description

2.55 Branch-Based

Collaboration diagram for Branch-Based:



Modules

- [Base and Policy Classes](#)

Classes

- [class `__gnu_pbds::basic_branch`](#)`< Key, Mapped, Tag, Node_Update, Policy_TI, _Alloc >`
- [class `__gnu_pbds::tree`](#)`< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`
- [class `__gnu_pbds::trie`](#)`< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`

Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

2.55.1 Detailed Description

2.56 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



Classes

- class `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`
Ordered-vector tree associative-container.
- class `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >`
Conditional destructor.
- class `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`
PATRICIA trie.
This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.
- class `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`
Red-Black tree.
This implementation uses an idea from the SGI STL (using a header node which is needed for efficient iteration).
- class `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`
Splay tree.

2.56.1 Detailed Description

2.57 List-Based

Collaboration diagram for List-Based:



Classes

- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`

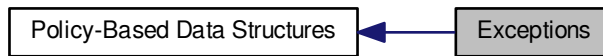
Macros

- `#define PB_DS_LU_BASE`

2.57.1 Detailed Description

2.58 Exceptions

Collaboration diagram for Exceptions:



Classes

- struct `__gnu_pbds::container_error`
Base class for exceptions.
- struct `__gnu_pbds::insert_error`
An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).
- struct `__gnu_pbds::join_error`
A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps).
- struct `__gnu_pbds::resize_error`
A container cannot be resized.

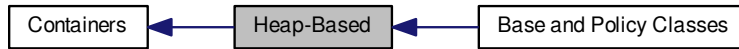
Functions

- void `__gnu_pbds::__throw_container_error()`
- void `__gnu_pbds::__throw_insert_error()`
- void `__gnu_pbds::__throw_join_error()`
- void `__gnu_pbds::__throw_resize_error()`

2.58.1 Detailed Description

2.59 Heap-Based

Collaboration diagram for Heap-Based:



Modules

- [Base and Policy Classes](#)

Classes

- class [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >](#)

Typedefs

- typedef `_Alloc` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::allocator_type](#)
- typedef `Cmp_Fn` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::cmp_fn](#)
- typedef `base_type::const_iterator` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::const_iterator](#)
- typedef `__rebind_va::const_pointer` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::const_pointer](#)
- typedef `__rebind_va::const_reference` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::const_reference](#)
- typedef `Tag` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::container_category](#)
- typedef `allocator_type::difference_type` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::difference_type](#)
- typedef `base_type::iterator` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::iterator](#)
- typedef `base_type::point_const_iterator` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::point_const_iterator](#)
- typedef `base_type::point_iterator` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::point_iterator](#)
- typedef `__rebind_va::pointer` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::pointer](#)
- typedef `__rebind_va::reference` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::reference](#)
- typedef `allocator_type::size_type` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::size_type](#)
- typedef `_Tv` [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::value_type](#)

Functions

- [__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue](#) (const cmp_fn &r_cmp_fn)
- template<typename It >
[__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue](#) (It first_it, It last_it)

- `template<typename It >`
`__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue` (`It first_it, It last_it, const cmp_fn &r_cmp_fn`)
- `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue` (`const priority_queue &other`)
- `priority_queue & __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::operator=` (`const priority_queue &other`)
- `void __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::swap` (`priority_queue &other`)

2.59.1 Detailed Description

2.59.2 Function Documentation

2.59.2.1 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue` (`const cmp_fn &r_cmp_fn`) [`inline`]

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

Definition at line 116 of file `priority_queue.hpp`.

2.59.2.2 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue` (`It first_it, It last_it`) [`inline`]

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 122 of file `priority_queue.hpp`.

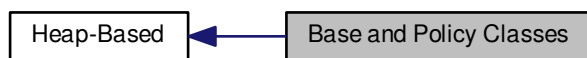
2.59.2.3 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue` (`It first_it, It last_it, const cmp_fn &r_cmp_fn`) [`inline`]

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object.

Definition at line 130 of file `priority_queue.hpp`.

2.60 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



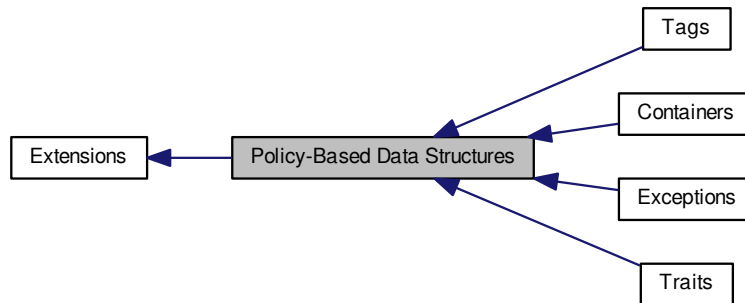
Classes

- class `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`

2.60.1 Detailed Description

2.61 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Modules

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

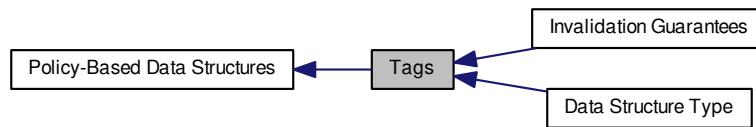
2.61.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

2.62 Tags

Collaboration diagram for Tags:



Modules

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

Classes

- struct [__gnu_pbds::trivial_iterator_tag](#)
A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.

Typedefs

- typedef void [__gnu_pbds::trivial_iterator_difference_type](#)

2.62.1 Detailed Description

2.62.2 Typedef Documentation

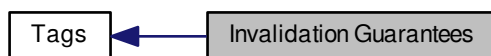
2.62.2.1 typedef void [__gnu_pbds::trivial_iterator_difference_type](#)

Prohibit moving trivial iterators.

Definition at line 79 of file tag_and_trait.hpp.

2.63 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



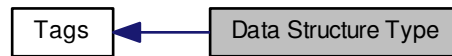
Classes

- struct [__gnu_pbds::basic_invalidation_guarantee](#)
- struct [__gnu_pbds::point_invalidation_guarantee](#)
- struct [__gnu_pbds::range_invalidation_guarantee](#)

2.63.1 Detailed Description

2.64 Data Structure Type

Collaboration diagram for Data Structure Type:



Classes

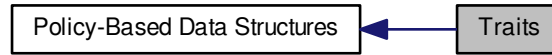
- struct [__gnu_pbds::associative_tag](#)
Basic associative-container.
- struct [__gnu_pbds::basic_branch_tag](#)
Basic branch structure.
- struct [__gnu_pbds::basic_hash_tag](#)
Basic hash structure.
- struct [__gnu_pbds::binary_heap_tag](#)
Binary-heap (array-based).
- struct [__gnu_pbds::binomial_heap_tag](#)
Binomial-heap.
- struct [__gnu_pbds::cc_hash_tag](#)
Collision-chaining hash.
- struct [__gnu_pbds::container_tag](#)
Base data structure tag.
- struct [__gnu_pbds::gp_hash_tag](#)
General-probing hash.
- struct [__gnu_pbds::list_update_tag](#)
List-update.
- struct [__gnu_pbds::ov_tree_tag](#)
Ordered-vector tree.
- struct [__gnu_pbds::pairing_heap_tag](#)
Pairing-heap.
- struct [__gnu_pbds::pat_trie_tag](#)
PATRICIA trie.
- struct [__gnu_pbds::priority_queue_tag](#)
Basic priority-queue.
- struct [__gnu_pbds::rb_tree_tag](#)
Red-black tree.
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
Redundant-counter binomial-heap.
- struct [__gnu_pbds::sequence_tag](#)
Basic sequence.

- struct [__gnu_pbds::splay_tree_tag](#)
Splay tree.
- struct [__gnu_pbds::string_tag](#)
Basic string container, inclusive of strings, ropes, etc.
- struct [__gnu_pbds::thin_heap_tag](#)
Thin heap.
- struct [__gnu_pbds::tree_tag](#)
Basic tree structure.
- struct [__gnu_pbds::trie_tag](#)
Basic trie structure.

2.64.1 Detailed Description

2.65 Traits

Collaboration diagram for Traits:



Classes

- struct [__gnu_pbds::container_traits< Cntnr >](#)
Container traits.
- struct [__gnu_pbds::container_traits_base< binary_heap_tag >](#)
Specialization, binary heap.
- struct [__gnu_pbds::container_traits_base< binomial_heap_tag >](#)
Specialization, binomial heap.
- struct [__gnu_pbds::container_traits_base< cc_hash_tag >](#)
Specialization, cc hash.
- struct [__gnu_pbds::container_traits_base< gp_hash_tag >](#)
Specialization, gp hash.
- struct [__gnu_pbds::container_traits_base< list_update_tag >](#)
Specialization, list update.
- struct [__gnu_pbds::container_traits_base< ov_tree_tag >](#)
Specialization, ov tree.
- struct [__gnu_pbds::container_traits_base< pairing_heap_tag >](#)
Specialization, pairing heap.
- struct [__gnu_pbds::container_traits_base< pat_trie_tag >](#)
Specialization, pat trie.
- struct [__gnu_pbds::container_traits_base< rb_tree_tag >](#)
Specialization, rb tree.
- struct [__gnu_pbds::container_traits_base< rc_binomial_heap_tag >](#)
Specialization, rc binomial heap.
- struct [__gnu_pbds::container_traits_base< splay_tree_tag >](#)
Specialization, splay tree.
- struct [__gnu_pbds::container_traits_base< thin_heap_tag >](#)
Specialization, thin heap.
- struct [__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >](#)
Binary search tree traits, primary template.
- struct [__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >](#)
Specialization.
- struct [__gnu_pbds::detail::no_throw_copies< Key, Mapped >](#)
Primary template.

- struct `__gnu_pbds::detail::no_throw_copies< Key, null_type >`
Specialization.
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th >`
Primary template for representation of stored data. Two types of data can be stored: value and hash.
- struct `__gnu_pbds::detail::stored_data< _Tv, null_type >`
Specialization for representation of stored data of just value type.
- struct `__gnu_pbds::detail::stored_hash< _Th >`
Stored hash.
- struct `__gnu_pbds::detail::stored_value< _Tv >`
Stored value.
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
Specialization, false.
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
Specialization, true.
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
Tree node metadata dispatch.
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
Tree traits.
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`
Specialization, false.
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
Specialization, true.
- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
Trie node metadata dispatch.
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
Specialization.
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`
- struct `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`
Type base dispatch.
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`
Traits for abstract types.
- struct `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`

A null node updator, indicating that no node updates are required.

- struct `__gnu_pbds::null_type`

Represents no type, or absence of type, for template tricks.

- struct `container_traits_base<_Tag>`

Primary template, container traits base.

- struct `tree_metadata_helper<Node_Update, _BTp>`

Tree metadata helper.

- struct `trie_metadata_helper<Node_Update, _BTp>`

Trie metadata helper.

- struct `type_base<Key, Mapped, _Alloc, Store_Hash>`

Primary template.

Variables

- static `null_type __gnu_pbds::detail::type_base<Key, null_type, _Alloc, false>::s_null_type`
- static `null_type __gnu_pbds::detail::type_base<Key, null_type, _Alloc, true>::s_null_type`

2.65.1 Detailed Description

2.66 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class `std::discard_block_engine<_RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m >`
A model of a linear congruential random number generator.
- class `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::random_device`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k >`
Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.

Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine`
`< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef
`linear_congruential_engine`
`< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` **`std::minstd_rand`**
- typedef
`linear_congruential_engine`
`< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` **`std::minstd_rand0`**
- typedef
`mersenne_twister_engine`
`< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` **`std::mt19937`**
- typedef
`mersenne_twister_engine`
`< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000-ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL >` **`std::mt19937_64`**
- typedef `discard_block_engine`
`< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef
`subtract_with_carry_engine`
`< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**

- `typedef discard_block_engine`
`< ranlux48_base, 389, 11 > std::ranlux48`
- `typedef`
`subtract_with_carry_engine`
`< uint_fast64_t, 48, 5, 12 > std::ranlux48_base`

Functions

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>`
`bool std::operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>`
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`
`bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

2.66.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.

Table 273: Random Number Generator Requirements

2.66.2 Typedef Documentation

2.66.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1527 of file random.h.

2.66.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

Definition at line 1521 of file random.h.

2.66.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo--Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1543 of file random.h.

2.66.2.4 `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xff7eee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1555 of file random.h.

2.66.3 Function Documentation

2.66.3.1 `template<typename UIntType, UIntType __a, UIntType __c, UIntType __m> bool std::operator!= (const std::linear_congruential_engine< UIntType, __a, __c, __m > & __lhs, const std::linear_congruential_engine< UIntType, __a, __c, __m > & __rhs) [inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 409 of file random.h.

2.66.3.2 `template<typename UIntType, size_t __w, size_t __n, size_t __m, size_t __r, UIntType __a, size_t __u, UIntType __d, size_t __s, UIntType __b, size_t __t, UIntType __c, size_t __l, UIntType __f> bool std::operator!= (const std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __lhs, const std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs) [inline]`

Compares two % mersenne_twister_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 641 of file random.h.

2.66.3.3 `template<typename UIntType, size_t __w, size_t __s, size_t __r> bool std::operator!= (const
std::subtract_with_carry_engine< UIntType, __w, __s, __r > & __lhs, const std::subtract_with_carry_engine< UIntType,
__w, __s, __r > & __rhs) [inline]`

Compares two % subtract_with_carry_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % subtract_with_carry_engine random number generator object.
<code>__rhs</code>	Another % subtract_with_carry_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 840 of file random.h.

2.66.3.4 `template<typename RandomNumberEngine, size_t __p, size_t __r> bool std::operator!= (const
std::discard_block_engine< RandomNumberEngine, __p, __r > & __lhs, const std::discard_block_engine<
RandomNumberEngine, __p, __r > & __rhs) [inline]`

Compares two discard_block_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A discard_block_engine random number generator object.
<code>__rhs</code>	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1062 of file random.h.

2.66.3.5 `template<typename RandomNumberEngine, size_t __w, typename UIntType > bool std::operator!= (const
std::independent_bits_engine< RandomNumberEngine, __w, UIntType > & __lhs, const
std::independent_bits_engine< RandomNumberEngine, __w, UIntType > & __rhs) [inline]`

Compares two independent_bits_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A independent_bits_engine random number generator object.
<code>__rhs</code>	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1258 of file random.h.

```
2.66.3.6  template<typename _RandomNumberEngine, size_t _k> bool std::operator!=( const std::shuffle_order_engine<
    _RandomNumberEngine, _k > & __lhs, const std::shuffle_order_engine< _RandomNumberEngine, _k > & __rhs )
    [inline]
```

Compares two shuffle_order_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1510 of file random.h.

```
2.66.3.7  template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >
    std::basic_ostream<_CharT, _Traits>& std::operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const
    std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x )
```

Inserts the current state of a independent_bits_engine random number generator engine `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A independent_bits_engine random number generator engine.

Returns

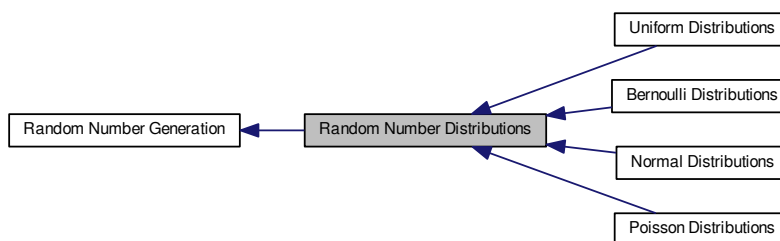
The output stream with the state of `__x` inserted or in an error state.

Definition at line 1277 of file random.h.

References `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::base()`.

2.67 Random Number Distributions

Collaboration diagram for Random Number Distributions:



Modules

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Uniform Distributions](#)

2.67.1 Detailed Description

2.68 Uniform Distributions

Collaboration diagram for Uniform Distributions:



Classes

- class `std::uniform_int_distribution<_IntType>`
Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.
- struct `std::uniform_int_distribution<_IntType>::param_type`
- class `std::uniform_real_distribution<_RealType>`
Uniform continuous distribution for random numbers.
- struct `std::uniform_real_distribution<_RealType>::param_type`

Functions

- template<typename _IntType>
`bool std::operator!= (const std::uniform_int_distribution<_IntType> &__d1, const std::uniform_int_distribution<_IntType> &__d2)`
- template<typename _IntType>
`bool std::operator!= (const std::uniform_real_distribution<_IntType> &__d1, const std::uniform_real_distribution<_IntType> &__d2)`
- template<typename _IntType, typename _CharT, typename _Traits>
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_int_distribution<_IntType> &)`
- template<typename _RealType, typename _CharT, typename _Traits>
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_real_distribution<_RealType> &)`
- template<typename _IntType, typename _CharT, typename _Traits>
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_int_distribution<_IntType> &)`
- template<typename _RealType, typename _CharT, typename _Traits>
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_real_distribution<_RealType> &)`

2.68.1 Detailed Description

2.68.2 Function Documentation

2.68.2.1 `template<typename _IntType> bool std::operator!=(const std::uniform_int_distribution< _IntType> & __d1, const std::uniform_int_distribution< _IntType> & __d2) [inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1825 of file random.h.

2.68.2.2 `template<typename _IntType> bool std::operator!=(const std::uniform_real_distribution< _IntType> & __d1, const std::uniform_real_distribution< _IntType> & __d2) [inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 2034 of file random.h.

2.68.2.3 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<<(std::basic_ostream< _CharT, _Traits> & __os, const std::uniform_int_distribution< _IntType> & __x)`

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1029 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.68.2.4 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<<(std::basic_ostream< _CharT, _Traits> & __os, const std::uniform_real_distribution< _RealType> & __x)`

Inserts a `uniform_real_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1088 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.68.2.5 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::uniform_int_distribution< _IntType> & __x)`

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1050 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution< _IntType>::param()`, and `std::skipws()`.

2.68.2.6 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::uniform_real_distribution< _RealType> & __x)`

Extracts a `uniform_real_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1112 of file `bits/random.tcc`.

References `std::ios_base::flags()`, `std::uniform_real_distribution< _RealType>::param()`, and `std::skipws()`.

2.69 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class `std::cauchy_distribution<_RealType>`
A cauchy_distribution random number distribution.
- struct `std::cauchy_distribution<_RealType>::param_type`
- class `std::chi_squared_distribution<_RealType>`
A chi_squared_distribution random number distribution.
- struct `std::chi_squared_distribution<_RealType>::param_type`
- class `std::fisher_f_distribution<_RealType>`
A fisher_f_distribution random number distribution.
- struct `std::fisher_f_distribution<_RealType>::param_type`
- class `std::gamma_distribution<_RealType>`
A gamma continuous distribution for random numbers.
- struct `std::gamma_distribution<_RealType>::param_type`
- class `std::lognormal_distribution<_RealType>`
A lognormal_distribution random number distribution.
- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::normal_distribution<_RealType>`
A normal continuous distribution for random numbers.
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::student_t_distribution<_RealType>`
A student_t_distribution random number distribution.
- struct `std::student_t_distribution<_RealType>::param_type`

Functions

- template<typename _RealType>
 bool `std::operator!=` (const `std::normal_distribution<_RealType>` &__d1, const `std::normal_distribution<_RealType>` &__d2)
- template<typename _RealType>
 bool `std::operator!=` (const `std::lognormal_distribution<_RealType>` &__d1, const `std::lognormal_distribution<_RealType>` &__d2)
- template<typename _RealType>
 bool `std::operator!=` (const `std::gamma_distribution<_RealType>` &__d1, const `std::gamma_distribution<_RealType>` &__d2)

- `template<typename _RealType >`
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)`

2.69.1 Detailed Description

2.69.2 Function Documentation

2.69.2.1 `template<typename _RealType > bool std::operator!= (const std::normal_distribution< _RealType > & __d1, const std::normal_distribution< _RealType > & __d2) [inline]`

Return true if two normal distributions are different.

Definition at line 2283 of file random.h.

2.69.2.2 `template<typename _RealType > bool std::operator!= (const std::lognormal_distribution< _RealType > & __d1, const std::lognormal_distribution< _RealType > & __d2) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2487 of file random.h.

2.69.2.3 `template<typename _RealType > bool std::operator!= (const std::gamma_distribution< _RealType > & __d1, const std::gamma_distribution< _RealType > & __d2) [inline]`

Return true if two gamma distributions are different.

Definition at line 2707 of file random.h.

2.69.2.4 `template<typename _RealType > bool std::operator!= (const std::chi_squared_distribution< _RealType > & __d1, const std::chi_squared_distribution< _RealType > & __d2) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2917 of file random.h.

2.69.2.5 `template<typename _RealType > bool std::operator!= (const std::cauchy_distribution< _RealType > & __d1, const std::cauchy_distribution< _RealType > & __d2) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 3084 of file random.h.

2.69.2.6 `template<typename _RealType> bool std::operator!=(const std::fisher_f_distribution< _RealType> & __d1, const std::fisher_f_distribution< _RealType> & __d2) [inline]`

Return true if two Fisher f distributions are diferent.

Definition at line 3340 of file random.h.

2.69.2.7 `template<typename _RealType> bool std::operator!=(const std::student_t_distribution< _RealType> & __d1, const std::student_t_distribution< _RealType> & __d2) [inline]`

Return true if two Student t distributions are different.

Definition at line 3553 of file random.h.

2.69.2.8 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< (std::basic_ostream< _CharT, _Traits> & __os, const std::cauchy_distribution< _RealType> & __x)`

Inserts a cauchy_distribution random number distribution __x into the output stream __os.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A cauchy_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

Definition at line 2273 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.69.2.9 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::cauchy_distribution< _RealType> & __x)`

Extracts a cauchy_distribution random number distribution __x from the input stream __is.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A cauchy_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

Definition at line 2297 of file bits/random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType>::param()`, and `std::skipws()`.

2.70 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class `std::bernoulli_distribution`
A Bernoulli random number distribution.
- struct `std::bernoulli_distribution::param_type`
- class `std::binomial_distribution<_IntType>`
A discrete binomial random number distribution.
- struct `std::binomial_distribution<_IntType>::param_type`
- class `std::geometric_distribution<_IntType>`
A discrete geometric random number distribution.
- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::negative_binomial_distribution<_IntType>`
A negative_binomial_distribution random number distribution.
- struct `std::negative_binomial_distribution<_IntType>::param_type`

Functions

- bool `std::operator!=` (const `std::bernoulli_distribution` &__d1, const `std::bernoulli_distribution` &__d2)
- template<typename `_IntType` >
bool `std::operator!=` (const `std::binomial_distribution<_IntType>` &__d1, const `std::binomial_distribution<_IntType>` &__d2)
- template<typename `_IntType` >
bool `std::operator!=` (const `std::geometric_distribution<_IntType>` &__d1, const `std::geometric_distribution<_IntType>` &__d2)
- template<typename `_IntType` >
bool `std::operator!=` (const `std::negative_binomial_distribution<_IntType>` &__d1, const `std::negative_binomial_distribution<_IntType>` &__d2)
- template<typename `_CharT`, typename `_Traits` >
`std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &__os, const `std::bernoulli_distribution` &__x)
- template<typename `_IntType`, typename `_CharT`, typename `_Traits` >
`std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &__os, const `std::geometric_distribution<_IntType>` &__x)

- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::bernoulli_distribution & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::geometric_distribution< _IntType`
`> & __x)`

2.70.1 Detailed Description

2.70.2 Function Documentation

2.70.2.1 `bool std::operator!= (const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2)`
`[inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3729 of file random.h.

2.70.2.2 `template<typename _IntType > bool std::operator!= (const std::binomial_distribution< _IntType > & __d1, const`
`std::binomial_distribution< _IntType > & __d2) [inline]`

Return true if two binomial distributions are different.

Definition at line 3994 of file random.h.

2.70.2.3 `template<typename _IntType > bool std::operator!= (const std::geometric_distribution< _IntType > & __d1, const`
`std::geometric_distribution< _IntType > & __d2) [inline]`

Return true if two geometric distributions have different parameters.

Definition at line 4163 of file random.h.

2.70.2.4 `template<typename _IntType > bool std::operator!= (const std::negative_binomial_distribution< _IntType > &`
`__d1, const std::negative_binomial_distribution< _IntType > & __d2) [inline]`

Return true if two negative binomial distributions are different.

Definition at line 4408 of file random.h.

2.70.2.5 `template<typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (`
`std::basic_ostream< _CharT, _Traits > & __os, const std::bernoulli_distribution & __x)`

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1150 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.70.2.6 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< (std::basic_ostream< _CharT, _Traits> & __os, const std::geometric_distribution< _IntType> & __x)`

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>geometric_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1232 of file `bits/random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.70.2.7 `template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::bernoulli_distribution & __x)`

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 3759 of file `random.h`.

References `std::bernoulli_distribution::param()`.

2.70.2.8 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::geometric_distribution< _IntType> & __x)`

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>geometric_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1256 of file `bits/random.tcc`.

References `std::ios_base::flags()`, `std::geometric_distribution< _IntType>::param()`, and `std::skipws()`.

2.71 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class `std::discrete_distribution< _IntType >`
A discrete_distribution random number distribution.
- struct `std::discrete_distribution< _IntType >::param_type`
- class `std::exponential_distribution< _RealType >`
An exponential continuous distribution for random numbers.
- struct `std::exponential_distribution< _RealType >::param_type`
- class `std::extreme_value_distribution< _RealType >`
A extreme_value_distribution random number distribution.
- struct `std::extreme_value_distribution< _RealType >::param_type`
- class `std::piecewise_constant_distribution< _RealType >`
A piecewise_constant_distribution random number distribution.
- struct `std::piecewise_constant_distribution< _RealType >::param_type`
- class `std::piecewise_linear_distribution< _RealType >`
A piecewise_linear_distribution random number distribution.
- struct `std::piecewise_linear_distribution< _RealType >::param_type`
- class `std::poisson_distribution< _IntType >`
A discrete Poisson random number distribution.
- struct `std::poisson_distribution< _IntType >::param_type`
- class `std::weibull_distribution< _RealType >`
A weibull_distribution random number distribution.
- struct `std::weibull_distribution< _RealType >::param_type`

Functions

- template<typename _IntType >
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- template<typename _RealType >
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- template<typename _RealType >
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`

- `template<typename _RealType >`
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`

2.71.1 Detailed Description

2.71.2 Function Documentation

2.71.2.1 `template<typename _IntType > bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2) [inline]`

Return true if two Poisson distributions are different.

Definition at line 4624 of file random.h.

2.71.2.2 `template<typename _RealType > bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4802 of file random.h.

2.71.2.3 `template<typename _RealType > bool std::operator!=(const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 5005 of file random.h.

2.71.2.4 `template<typename _RealType > bool std::operator!=(const std::extreme_value_distribution< _RealType > & __d1, const std::extreme_value_distribution< _RealType > & __d2) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 5208 of file random.h.

2.71.2.5 `template<typename _IntType > bool std::operator!=(const std::discrete_distribution< _IntType > & __d1, const std::discrete_distribution< _IntType > & __d2) [inline]`

Return true if two discrete distributions have different parameters.

Definition at line 5468 of file random.h.

2.71.2.6 `template<typename _RealType > bool std::operator!=(const std::piecewise_constant_distribution< _RealType > & __d1, const std::piecewise_constant_distribution< _RealType > & __d2) [inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5735 of file random.h.

2.71.2.7 `template<typename _RealType > bool std::operator!=(const std::piecewise_linear_distribution< _RealType > & __d1, const std::piecewise_linear_distribution< _RealType > & __d2) [inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 6005 of file random.h.

2.71.2.8 `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::exponential_distribution< _RealType > & __x)`

Inserts a `exponential_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>exponential_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1881 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.71.2.9 `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::weibull_distribution< _RealType > & __x)`

Inserts a `weibull_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A weibull_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2672 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.71.2.10 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< (std::basic_ostream< _CharT, _Traits> & __os, const std::extreme_value_distribution< _RealType> & __x)`

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2748 of file bits/random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.71.2.11 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::exponential_distribution< _RealType> & __x)`

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>exponential_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1904 of file bits/random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution< _RealType>::param()`, and `std::skipws()`.

2.71.2.12 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits> & std::operator>> (std::basic_istream< _CharT, _Traits> & __is, std::weibull_distribution< _RealType> & __x)`

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A weibull_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2696 of file bits/random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution<_RealType>::param()`, and `std::skipws()`.

```
2.71.2.13 template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> &
std::operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::extreme_value_distribution<_RealType>
> & __x )
```

Extracts a `extreme_value_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2772 of file bits/random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution<_RealType>::param()`, and `std::skipws()`.

2.72 Random Number Utilities

Collaboration diagram for Random Number Utilities:



Classes

- class [std::seed_seq](#)

The `seed_seq` class generates sequences of seeds for random number generators.

2.72.1 Detailed Description

3 Namespace Documentation

3.1 __gnu_cxx Namespace Reference

Namespaces

- namespace [__detail](#)
- namespace [typelist](#)

Classes

- struct [__alloc_traits](#)
Uniform interface to C++98 and C++0x allocators.
- struct [__common_pool_policy](#)
Policy for shared __pool objects.
- class [__mt_alloc](#)
This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.
- class [__mt_alloc_base](#)
Base class for _Tp dependent member functions.
- struct [__per_type_pool_policy](#)
Policy for individual __pool objects.
- class [__pool< false >](#)
Specialization for single thread.
- class [__pool< true >](#)
Specialization for thread enabled, via gthreads.h.
- class [__pool_alloc](#)
Allocator using a memory pool with a single lock.
- class [__pool_alloc_base](#)
Base class for __pool_alloc.
- struct [__pool_base](#)
Base class for pool object.
- class [__rc_string_base](#)
- class [__scoped_lock](#)
Scoped lock idiom.
- class [__versa_string](#)
*Template class __versa_string.
Data structure managing sequences of characters and character-like objects.*
- struct [_Caster](#)
- struct [_Char_types](#)
Mapping from character type to associated types.
- class [_ExtPtr_allocator](#)
*An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.*
- struct [_Invalid_type](#)
- class [_Pointer_adapter](#)
- class [_Relative_pointer_impl](#)

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

- class [_Relative_pointer_impl< const_Tp >](#)
- class [_Std_pointer_impl](#)

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

- struct [_Unqualified_type](#)
- struct [annotate_base](#)

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

- class [array_allocator](#)

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

- class [array_allocator_base](#)

Base class.

- class [binary_compose](#)

An *SGL extension*.

- class [bitmap_allocator](#)

Bitmap Allocator, primary template.

- struct [char_traits](#)

Base class used to implement `std::char_traits`.

- struct [character](#)

A POD class that serves as a character abstraction class.

- struct [condition_base](#)

Base struct for condition policy.

- struct [constant_binary_fun](#)

An *SGL extension*.

- struct [constant_unary_fun](#)

An *SGL extension*.

- struct [constant_void_fun](#)

An *SGL extension*.

- class [debug_allocator](#)

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- class [enc_filebuf](#)

class `enc_filebuf`.

- struct [encoding_char_traits](#)

`encoding_char_traits`

- class [encoding_state](#)

Extension to use `iconv` for dealing with character encodings.

- struct [forced_error](#)

Thrown by exception safety machinery.

- class [free_list](#)

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

- class [hash_map](#)
- class [hash_multimap](#)
- class [hash_multiset](#)
- class [hash_set](#)
- struct [limit_condition](#)

Base class for incremental control and throw.

- class `malloc_allocator`
*An allocator that uses malloc.
This is precisely the allocator defined in the C++ Standard.*
- class `new_allocator`
*An allocator that uses global new, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
- struct `project1st`
An SGI extension .
- struct `project2nd`
An SGI extension .
- struct `random_condition`
Base class for random probability control and throw.
- struct `rb_tree`
- class `recursive_init_error`
*Exception thrown by `__cxa_guard_acquire`.
6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `rope`
- struct `select1st`
An SGI extension .
- struct `select2nd`
An SGI extension .
- class `slist`
- class `stdio_filebuf`
*Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `stdio_sync_filebuf`
*Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `subtractive_rng`
- struct `temporary_buffer`
- class `throw_allocator_base`
*Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.
Note: Deallocate not allowed to throw.*
- struct `throw_allocator_limit`
Allocator throwing via limit condition.
- struct `throw_allocator_random`
Allocator throwing via random condition.
- struct `throw_value_base`
Class with exception generation control. Intended to be used as a `value_type` in templated code.
- struct `throw_value_limit`
Type throwing via limit condition.
- struct `throw_value_random`
Type throwing via random condition.
- class `unary_compose`
An SGI extension .

Typedefs

- typedef void(* **__destroy_handler**)(void *)
- typedef **__versa_string**< char,
[std::char_traits](#)< char >
, [std::allocator](#)< char >
, **__rc_string_base** > **__rc_string**
- typedef **__vstring** **__sso_string**
- typedef **__versa_string**
< char16_t, [std::char_traits](#)
< char16_t >, [std::allocator](#)
< char16_t >, **__rc_string_base** > **__u16rc_string**
- typedef **__u16vstring** **__u16sso_string**
- typedef **__versa_string**< char16_t > **__u16vstring**
- typedef **__versa_string**
< char32_t, [std::char_traits](#)
< char32_t >, [std::allocator](#)
< char32_t >, **__rc_string_base** > **__u32rc_string**
- typedef **__u32vstring** **__u32sso_string**
- typedef **__versa_string**< char32_t > **__u32vstring**
- typedef **__versa_string**< char > **__vstring**
- typedef **__versa_string**
< wchar_t, [std::char_traits](#)
< wchar_t >, [std::allocator](#)
< wchar_t >, **__rc_string_base** > **__wrc_string**
- typedef **__wvstring** **__wsso_string**
- typedef **__versa_string**< wchar_t > **__wvstring**
- typedef [rope](#)< char > **crope**
- typedef [rope](#)< wchar_t > **wrope**

Enumerations

- enum { **_S_num_primes** }
- enum **_Lock_policy** { **_S_single**, **_S_mutex**, **_S_atomic** }

Functions

- static void **__atomic_add_single** (_Atomic_word * __mem, int __val)
- else **__atomic_add_single** (__mem, __val)
- _Atomic_word **__attribute__**((__unused__)) **__exchange_and_add**(volatile _Atomic_word *
- template<class _Tp >
void **__aux_require_boolean_expr** (const _Tp & __t)
- template<typename _ToType , typename _FromType >
_ToType **__const_pointer_cast** (const _FromType & __arg)
- template<typename _ToType , typename _FromType >
_ToType **__const_pointer_cast** (_FromType * __arg)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
[pair](#)< _InputIterator,
_OutputIterator > **__copy_n** (_InputIterator __first, _Size __count, _OutputIterator __result, [input_iterator_tag](#))

- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`
`pair<_RAIterator,`
`_OutputIterator > __copy_n (_RAIterator __first, _Size __count, _OutputIterator __result, random_access_`
`iterator_tag)`
- `template<typename _InputIterator, typename _Distance >`
`void __distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`void __distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_`
`access_iterator_tag)`
- `template<typename _ToType, typename _FromType >`
`_ToType __dynamic_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __dynamic_pointer_cast (_FromType *__arg)`
- `void __error_type_must_be_a_signed_integer_type ()`
- `void __error_type_must_be_an_integer_type ()`
- `void __error_type_must_be_an_unsigned_integer_type ()`
- `static _Atomic_word __exchange_and_add_single (_Atomic_word *__mem, int __val)`
- `else return __exchange_and_add_single (__mem, __val)`
- `template<class _Concept >`
`void __function_requires ()`
- `template<typename _Type >`
`bool __is_null_pointer (_Type *__ptr)`
- `template<typename _Type >`
`bool __is_null_pointer (_Type)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`
`_InputIterator2 __last2)`
- `int __lexicographical_compare_3way (const unsigned char *__first1, const unsigned char *__last1, const un-`
`signed char *__first2, const unsigned char *__last2)`
- `int __lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const`
`char *__last2)`
- `template<typename _Tp >`
`const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference __mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp __power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`
`_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator`
`__out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`
`_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator`
`__out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (_FromType *__arg)`
- `_Slist_node_base * __slist_make_link (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)`
- `_Slist_node_base * __slist_previous (_Slist_node_base *__head, const _Slist_node_base *__node)`

- `const _Slist_node_base * __slist_previous` (`const _Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `_Slist_node_base * __slist_reverse` (`_Slist_node_base * __node`)
- `size_t __slist_size` (`_Slist_node_base * __node`)
- `void __slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __before_first`, `_Slist_node_base * __before_last`)
- `void __slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __head`)
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast` (`const _FromType & __arg`)
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast` (`_FromType * __arg`)
- `size_t __stl_hash_string` (`const char * __s`)
- `unsigned long __stl_next_prime` (`unsigned long __n`)
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`
`_Ret __stoa` (`_TRet(* __convf)(const _CharT *, _CharT **, _Base...)`, `const char * __name`, `const _CharT * __str`, `std::size_t * __idx`, `_Base... __base`)
- `void __throw_concurrency_lock_error` ()
- `void __throw_concurrency_unlock_error` ()
- `void __throw_forced_error` ()
- `template<typename _String, typename _CharT = typename _String::value_type>`
`_String __to_xstring` (`int(* __convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list)`, `std::size_t __n`, `const _CharT * __fmt...`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`, `std::input_iterator_tag`)
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`
`pair< _RandomAccessIter, _ForwardIter > __uninitialized_copy_n` (`_RandomAccessIter __first`, `_Size __count`, `_ForwardIter __result`, `std::random_access_iterator_tag`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`, `_Allocator __alloc`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`, `std::allocator<_Tp>`)
- `void __verbose_terminate_handler` ()
- `size_t __Bit_scan_forward` (`size_t __num`)
- `template<typename _ForwardIterator, typename _Allocator >`
`void __Destroy_const` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Allocator __alloc`)
- `template<typename _ForwardIterator, typename _Tp >`
`void __Destroy_const` (`_ForwardIterator __first`, `_ForwardIterator __last`, `allocator<_Tp>`)
- `template<class _CharT, class _Traits >`
`void __Rope_fill` (`basic_ostream< _CharT, _Traits > & __o`, `size_t __n`)
- `template<class _CharT >`
`bool __Rope_is_simple` (`_CharT *`)
- `bool __Rope_is_simple` (`char *`)
- `bool __Rope_is_simple` (`wchar_t *`)
- `template<class _Rope_iterator >`
`void __Rope_rotate` (`_Rope_iterator __first`, `_Rope_iterator __middle`, `_Rope_iterator __last`)

- `template<class _CharT >`
`void _S_cond_store_eos (_CharT &)`
- `void _S_cond_store_eos (char &__c)`
- `void _S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`
`_CharT _S_eos (_CharT *)`
- `template<class _CharT >`
`bool _S_is_basic_char_type (_CharT *)`
- `bool _S_is_basic_char_type (char *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`
`bool _S_is_one_byte_char_type (_CharT *)`
- `bool _S_is_one_byte_char_type (char *)`
- `template<class _Operation1, class _Operation2 >`
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator, typename _Distance >`
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`
`_Tp identity_element (std::plus< _Tp >)`
- `template<class _Tp >`
`_Tp identity_element (std::multiplies< _Tp >)`
- `static _Atomic_word int __val if (__pthread_active_p()) return __exchange_and_add(__mem`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Tp >`
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

- `template<typename _Tp, typename _Array >`
`bool operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp >`
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFcn, class _EqKey, class _Alloc >`
`bool operator!= (const hash_map< _Key, _Tp, _HashFcn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFcn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const __Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (_Tp1 __lhs, const __Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const __Pointer_adapter< _Tp1 > &__lhs, const __Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const __Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool operator!= (int __lhs, const __Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const __Pointer_adapter< _Tp > &__lhs, const __Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp, typename _Cond >`
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Tp, typename _Poolp >`
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<class _Tp, class _Alloc >`
`bool operator!= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`

- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT * __right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT
__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa-
string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_-
string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string<
_CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base >
&&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT
*__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)`
- `template<class _CharT, class _Alloc >
rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >
rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >
rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >
_Rope_const_iterator< _CharT,
_Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >
ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,
_Alloc > &__y)`
- `template<class _CharT, class _Alloc >
_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >
ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__-
__y)`
- `template<typename _Cond >
throw_value_base< _Cond > operator- (const throw_value_base< _Cond > &__a, const throw_value_base<
_Cond > &__b)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >
auto operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR,
_Container > &__rhs) -> decltype(__lhs.base()-__rhs.base())`

- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator,`
`_Container >::difference_type operator- (const __normal_iterator< _Iterator, _Container > &__lhs, const __`
`normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename V, typename I, typename S >`
`bool operator< (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,`
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (_Tp1 __lhs, const Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const Pointer_adapter< _Tp1 > &__lhs, const Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Cond >`
`bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`
`bool operator< (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator< (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _Iterator-`
`R, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator< (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator,`
`_Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT,`
`_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT`
`, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_-`
`mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3,`
`__msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution<`
`_RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const Pointer_adapter< _StoreT >`
`&__p)`

- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_`
`distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _Real-`
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType >`
`&__x)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >`
`bool operator<= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _`
`IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator,`
`_Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _Char-`
`T, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename V, typename I, typename S >`
`bool operator== (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<typename _Tp >`
`bool operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp, typename _Array >`
`bool operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp >`
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`
`bool operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`

- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Cond >`
`bool operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp >`
`bool operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<size_t _Dimen, typename _RealType >`
`bool operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _Tp, typename _Cond >`
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Tp, typename _Poolp >`
`bool operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<class _Tp, class _Alloc >`
`bool operator== (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::is_char< _CharT >::value, bool >`
`::__type operator== (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >`
`bool operator> (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`

- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp >`
`bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >`
`bool operator>= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >`

- ```
std::basic_istream< _CharT,
_Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_
twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, _
__parity1, __parity2, __parity3, __parity4 > &__x)
```
- template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, \_\_gnu\_cxx::beta\_distribution< \_Real-  
Type > &\_\_x)
  - template<size\_t \_Dimen, typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, \_\_gnu\_cxx::normal\_mv\_distribution<  
\_Dimen, \_RealType > &\_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, rice\_distribution< \_RealType > &\_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, nakagami\_distribution< \_RealType >  
&\_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, pareto\_distribution< \_RealType > &\_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, k\_distribution< \_RealType > &\_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, arcsine\_distribution< \_RealType > &\_  
\_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & **operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, hoyt\_distribution< \_RealType > &\_\_x)
  - template<typename \_Tp, typename \_Integer, typename \_MonoidOperation >  
\_Tp [power](#) (\_Tp \_\_x, \_Integer \_\_n, \_MonoidOperation \_\_monoid\_op)
  - template<typename \_Tp, typename \_Integer >  
\_Tp [power](#) (\_Tp \_\_x, \_Integer \_\_n)
  - template<typename \_InputIterator, typename \_RandomAccessIterator >  
\_RandomAccessIterator [random\\_sample](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_RandomAccessIterator \_  
\_out\_first, \_RandomAccessIterator \_\_out\_last)
  - template<typename \_InputIterator, typename \_RandomAccessIterator, typename \_RandomNumberGenerator >  
\_RandomAccessIterator [random\\_sample](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_RandomAccessIterator \_  
\_out\_first, \_RandomAccessIterator \_\_out\_last, \_RandomNumberGenerator &\_\_rand)
  - template<typename \_ForwardIterator, typename \_OutputIterator, typename \_Distance >  
\_OutputIterator [random\\_sample\\_n](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_OutputIterator \_\_out, const  
\_Distance \_\_n)
  - template<typename \_ForwardIterator, typename \_OutputIterator, typename \_Distance, typename \_RandomNumberGenerator >  
\_OutputIterator [random\\_sample\\_n](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_OutputIterator \_\_out, const  
\_Distance \_\_n, \_RandomNumberGenerator &\_\_rand)
  - void **rotate** (\_Rope\_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> \_\_first, \_Rope\_iterator< char, \_\_S-  
TL\_DEFAULT\_ALLOCATOR(char)> \_\_middle, \_Rope\_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)>  
\_\_last)
  - template<typename \_Tp >  
void **swap** (\_ExtPtr\_allocator< \_Tp > &\_\_larg, \_ExtPtr\_allocator< \_Tp > &\_\_rarg)

- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Cond >`  
`void swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`  
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`
- `template<class _Tp, class _Alloc >`  
`void swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`void swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`
- `_Atomic_word int throw ()`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`

## Variables

- `static const _Lock_policy __default_lock_policy`
- `static _Atomic_word int __val __val`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

### 3.1.1 Detailed Description

GNU extensions for public use.

### 3.1.2 Function Documentation

#### 3.1.2.1 `template<typename _ToType, typename _FromType > _ToType __gnu_cxx::__static_pointer_cast ( const _FromType & __arg )` `[inline]`

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 68 of file `cast.h`.

3.1.2.2 `template<typename _ToType , typename _FromType > _ToType __gnu_cxx::__static_pointer_cast ( _FromType * __arg )`  
`[inline]`

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 96 of file `cast.h`.

3.1.2.3 `size_t __gnu_cxx::Bit_scan_forward ( size_t __num )` `[inline]`

Generic Version of the `bsf` instruction.

Definition at line 513 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

3.1.2.4 `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )` `[inline]`

Test difference of two strings.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2279 of file `vstring.h`.

3.1.2.5 `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`  
`[inline]`

Test difference of C string and string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2292 of file `vstring.h`.

3.1.2.6 `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs )`  
`[inline]`

Test difference of string and C string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

## Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2305 of file `vstring.h`.

**3.1.2.7** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate two strings.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 180 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**3.1.2.8** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate C string and string.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 193 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**3.1.2.9** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( _CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate character and string.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 210 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**3.1.2.10** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT * __rhs )`

Concatenate string and C string.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 223 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**3.1.2.11** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, _CharT __rhs )`

Concatenate string and character.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 240 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

3.1.2.12 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string precedes string.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

#### Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2319 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.13 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string precedes C string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

#### Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2332 of file `vstring.h`.

3.1.2.14 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string precedes string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2345 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.15 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't follow string.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

#### Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2399 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.16 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't follow C string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

#### Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2412 of file `vstring.h`.

3.1.2.17 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string doesn't follow string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2425 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.18 `template<typename _Tp> bool __gnu_cxx::operator==( const _Pointer_adapter< _Tp> & __lhs, const _Pointer_adapter< _Tp> & __rhs ) [inline]`

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

Definition at line 529 of file `pointer.h`.

3.1.2.19 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator==( const __versa_string< _CharT, _Traits, _Alloc, _Base> & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base> & __rhs ) [inline]`

Test equivalence of two strings.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

#### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2228 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

3.1.2.20 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator==( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base> & __rhs ) [inline]`

Test equivalence of C string and string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2252 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

3.1.2.21 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator==( const __versa_string< _CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT * __rhs ) [inline]`

Test equivalence of string and C string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2265 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.22 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string follows string.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2359 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.23 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs
) [inline]
```

Test if string follows C string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2372 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.24 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator> (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs
) [inline]
```

Test if C string follows string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2385 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.25 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
 class _Base> bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const
 __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't precede string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2439 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.26 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
 _Base> bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs
) [inline]
```

Test if string doesn't precede C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2452 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.27 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
 _Base> bool __gnu_cxx::operator>= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs
) [inline]
```

Test if C string doesn't precede string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

## Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2465 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.28 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Swap contents of two strings.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2479 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

3.2 `__gnu_cxx::__detail` Namespace Reference

## Classes

- class [\\_\\_mini\\_vector](#)  
*`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.*
- class [\\_Bitmap\\_counter](#)  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class [\\_Ffit\\_finder](#)  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*

## Enumerations

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

## Functions

- void [\\_\\_bit\\_allocate](#) (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- void [\\_\\_bit\\_free](#) (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator [\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
size\_t [\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
size\_t [\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)

## 3.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

## 3.2.2 Function Documentation

**3.2.2.1** `void __gnu_cxx::detail::_bit_allocate ( size_t * __pmap, size_t __pos ) throw () [inline]`

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 488 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

**3.2.2.2** `void __gnu_cxx::detail::_bit_free ( size_t * __pmap, size_t __pos ) throw () [inline]`

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 499 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`.

**3.2.2.3** `template<typename AddrPair> size_t __gnu_cxx::detail::_num_bitmaps ( AddrPair __ap ) [inline]`

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 276 of file `bitmap_allocator.h`.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`, and `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`.

**3.2.2.4** `template<typename AddrPair> size_t __gnu_cxx::detail::_num_blocks ( AddrPair __ap ) [inline]`

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 268 of file `bitmap_allocator.h`.

Referenced by `__num_bitmaps()`.

3.3 `__gnu_cxx::typelist` Namespace Reference

## Functions

- `template<typename Fn, typename Typelist>`  
`void apply (Fn &, Typelist)`
- `template<typename Gn, typename Typelist>`  
`void apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV>`  
`void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist>`  
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV>`  
`void apply_generator (Fn &fn, TypelistT, TypelistV)`

### 3.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

### 3.3.2 Function Documentation

3.3.2.1 `template<typename Gn , typename Typelist > void __gnu_cxx::typelist::apply_generator ( Gn & , Typelist )`

Apply all typelist types to generator functor.

## 3.4 `__gnu_debug` Namespace Reference

### Classes

- class [\\_After\\_nth\\_from](#)
- struct [\\_BeforeBeginHelper](#)
- class [\\_Equal\\_to](#)
- class [\\_Not\\_equal\\_to](#)
- class [\\_Safe\\_iterator](#)  
*Safe iterator wrapper.*
- class [\\_Safe\\_iterator\\_base](#)  
*Basic functionality for a safe iterator.*
- class [\\_Safe\\_local\\_iterator](#)  
*Safe iterator wrapper.*
- class [\\_Safe\\_local\\_iterator\\_base](#)  
*Basic functionality for a safe iterator.*
- class [\\_Safe\\_sequence](#)  
*Base class for constructing a safe sequence type that tracks iterators that reference it.*
- class [\\_Safe\\_sequence\\_base](#)  
*Base class that supports tracking of iterators that reference a sequence.*
- class [\\_Safe\\_unordered\\_container](#)  
*Base class for constructing a safe unordered container type that tracks iterators that reference it.*
- class [\\_Safe\\_unordered\\_container\\_base](#)  
*Base class that supports tracking of local iterators that reference an unordered container.*
- class [basic\\_string](#)  
*Class `std::basic_string` with safety/checking/debug instrumentation.*

### Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

## Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,  
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move_`  
`assign`,  
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs` }
- enum `_Distance_precision` { `__dp_equality`, `__dp_sign`, `__dp_exact` }

## Functions

- template<typename `_Iterator` >  
`_Siter_base`< `_Iterator` >  
`::iterator_type` `__base` (`_Iterator` \_\_it)
- template<typename `_Iterator` >  
bool `__check_dereferenceable` (`_Iterator` &)
- template<typename `_Tp` >  
bool `__check_dereferenceable` (const `_Tp` \* \_\_ptr)
- template<typename `_Iterator` , typename `_Sequence` >  
bool `__check_dereferenceable` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_x)
- template<typename `_ForwardIterator` , typename `_Tp` >  
bool `__check_partitioned_lower` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_value)
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_Pred` >  
bool `__check_partitioned_lower` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_value, `_Pred` \_\_pred)
- template<typename `_ForwardIterator` , typename `_Tp` >  
bool `__check_partitioned_lower_aux` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_value, `std::forward_iterator_tag`)
- template<typename `_Iterator` , typename `_Sequence` , typename `_Tp` >  
bool `__check_partitioned_lower_aux` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_first, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_last, const `_Tp` & \_\_value, `std::random_access_iterator_tag` \_\_tag)
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_Pred` >  
bool `__check_partitioned_lower_aux` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_value, `_Pred` \_\_pred, `std::forward_iterator_tag`)
- template<typename `_Iterator` , typename `_Sequence` , typename `_Tp` , typename `_Pred` >  
bool `__check_partitioned_lower_aux` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_first, const `_Safe_iterator`< `_Iterator`, `_Sequence` > & \_\_last, const `_Tp` & \_\_value, `_Pred` \_\_pred, `std::random_access_iterator_tag` \_\_tag)
- template<typename `_ForwardIterator` , typename `_Tp` >  
bool `__check_partitioned_upper` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_value)
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_Pred` >  
bool `__check_partitioned_upper` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_value, `_Pred` \_\_pred)

- `template<typename _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_upper_aux (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value,`  
`std::forward_iterator_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Tp >`  
`bool __check_partitioned_upper_aux (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_`  
`iterator< _Iterator, _Sequence > &__last, const _Tp &__value, std::random_access_iterator_tag __tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __check_partitioned_upper_aux (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value,`  
`_Pred __pred, std::forward_iterator_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Tp, typename _Pred >`  
`bool __check_partitioned_upper_aux (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_`  
`iterator< _Iterator, _Sequence > &__last, const _Tp &__value, _Pred __pred, std::random_access_iterator_tag`  
`__tag)`
- `template<typename _Iterator >`  
`bool __check_singular (_Iterator &)`
- `template<typename _Tp >`  
`bool __check_singular (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const _Safe_iterator_base *__x)`
- `template<typename _InputIterator >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __check_sorted_aux (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _`  
`Iterator, _Sequence > &__last, std::random_access_iterator_tag __tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_`  
`iterator_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Predicate >`  
`bool __check_sorted_aux (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _`  
`Iterator, _Sequence > &__last, _Predicate __pred, std::random_access_iterator_tag __tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &,`  
`_Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred,`  
`std::__true_type)`

- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_check\_sorted\_set\_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * \_\_check\_string (const _CharT * __s, const _Integer & __n \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _CharT >`  
`const _CharT * \_\_check\_string (const _CharT * __s)`
- `template<typename _InputIterator >`  
`_InputIterator \_\_check\_valid\_range (const _InputIterator & __first, const _InputIterator & __last \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _Iterator1, typename _Iterator2 >`  
`std::pair< typename  
std::iterator_traits  
< _Iterator1 >  
::difference_type,  
\_Distance\_precision > \_\_get\_distance (const _Iterator1 & __lhs, const _Iterator2 & __rhs, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator1, typename _Iterator2 >`  
`std::pair< typename  
std::iterator_traits  
< _Iterator1 >  
::difference_type,  
\_Distance\_precision > \_\_get\_distance (const _Iterator1 & __lhs, const _Iterator2 & __rhs, std::forward\_iterator\_tag)`
- `template<typename _Iterator1, typename _Iterator2 >`  
`std::pair< typename  
std::iterator_traits  
< _Iterator1 >  
::difference_type,  
\_Distance\_precision > \_\_get\_distance (const _Iterator1 & __lhs, const _Iterator2 & __rhs)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range (const _InputIterator & __first, const _InputIterator & __last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence > & __first, const \_Safe\_iterator< _Iterator, _Sequence > & __last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_valid\_range (const \_Safe\_local\_iterator< _Iterator, _Sequence > & __first, const \_Safe\_local\_iterator< _Iterator, _Sequence > & __last)`
- `template<typename _Integral >`  
`bool \_\_valid\_range\_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux (const _InputIterator & __first, const _InputIterator & __last, std::__false_type)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_valid\_range\_aux2 (const _RandomAccessIterator & __first, const _RandomAccessIterator & __last, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux2 (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT,  
_Traits > & getline (std::basic\_istream< _CharT, _Traits > & __is, basic\_string< _CharT, _Traits, _Allocator >  
& __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT,  
_Traits > & getline (std::basic\_istream< _CharT, _Traits > & __is, basic\_string< _CharT, _Traits, _Allocator >  
& __str)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const \_Safe\_local\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_local\_iterator< _`  
`IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const \_Safe\_local\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_local\_iterator< _`  
`Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _`  
`Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _`  
`Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`\_Safe\_iterator< _Iterator,`  
`_Sequence > operator+ (typename \_Safe\_iterator< _Iterator, _Sequence >::difference_type __n, const \_Safe-`  
`\_iterator< _Iterator, _Sequence > &__i)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits,`  
`_Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT,`  
`_Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits,`  
`_Allocator > operator+ (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits,`  
`_Allocator > operator+ (_CharT __lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits,`  
`_Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits,`  
`_Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`\_Safe\_iterator< _IteratorL,`  
`_Sequence >::difference_type operator- (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe-`  
`\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`\_Safe\_iterator< _Iterator,`  
`_Sequence >::difference_type operator- (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe-`  
`\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _`  
`Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _`  
`Sequence > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void swap (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

### 3.4.1 Detailed Description

GNU debug classes for public use.

### 3.4.2 Enumeration Type Documentation

#### 3.4.2.1 enum \_\_gnu\_debug::Distance\_precision

The precision to which we can calculate the distance between two iterators.

Definition at line 70 of file `safe_iterator.h`.

### 3.4.3 Function Documentation

#### 3.4.3.1 template<typename \_Iterator > \_Siter\_base<\_Iterator>::iterator\_type \_\_gnu\_debug::\_\_base ( \_Iterator \_\_it ) [inline]

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__valid_range` function thanks to the `<` operator.

Definition at line 546 of file `functions.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::M_before_dereferenceable()`, `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, `std::nouppercase()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

3.4.3.2 `template<typename _Iterator > bool __gnu_debug::__check_dereferenceable ( _Iterator & ) [inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 70 of file functions.h.

3.4.3.3 `template<typename _Tp > bool __gnu_debug::__check_dereferenceable ( const _Tp * __ptr ) [inline]`

Non-NULL pointers are dereferenceable.

Definition at line 76 of file functions.h.

3.4.3.4 `template<typename _Iterator, typename _Sequence > bool __gnu_debug::__check_dereferenceable ( const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 82 of file functions.h.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_dereferenceable()`.

3.4.3.5 `template<typename _Tp > bool __gnu_debug::__check_singular ( const _Tp * __ptr ) [inline]`

Non-NULL pointers are nonsingular.

Definition at line 57 of file functions.h.

3.4.3.6 `template<typename _Iterator, typename _Sequence > bool __gnu_debug::__check_singular ( const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 63 of file functions.h.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

3.4.3.7 `bool __gnu_debug::__check_singular_aux ( const _Safe_iterator_base * __x ) [inline]`

Iterators that derive from `_Safe_iterator_base` but that aren't `_Safe_iterators` can be determined singular or non-singular via `_Safe_iterator_base`.

Definition at line 64 of file safe\_iterator.h.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

3.4.3.8 `template<typename _CharT, typename _Integer > const _CharT* __gnu_debug::__check_string ( const _CharT * __s, const _Integer & __n __attribute__((unused)) ) [inline]`

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 168 of file functions.h.

3.4.3.9 `template<typename _CharT > const _CharT* __gnu_debug::__check_string ( const _CharT * __s ) [inline]`

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 180 of file functions.h.

3.4.3.10 `template<typename _Iterator1, typename _Iterator2> std::pair<typename std::iterator_traits<_Iterator1>::difference_type, _Distance_precision> __gnu_debug::__get_distance ( const _Iterator1 & __lhs, const _Iterator2 & __rhs, std::random_access_iterator_tag ) [inline]`

Determine the distance between two iterators with some known precision.

Definition at line 83 of file `safe_iterator.h`.

References `std::make_pair()`.

3.4.3.11 `template<typename _InputIterator> bool __gnu_debug::__valid_range ( const _InputIterator & __first, const _InputIterator & __last ) [inline]`

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `_InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 131 of file `functions.h`.

References `__valid_range_aux()`.

3.4.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::__valid_range ( const _Safe_iterator<_Iterator, _Sequence> & __first, const _Safe_iterator<_Iterator, _Sequence> & __last ) [inline]`

Safe iterators know how to check if they form a valid range.

Definition at line 140 of file `functions.h`.

3.4.3.13 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::__valid_range ( const _Safe_local_iterator<_Iterator, _Sequence> & __first, const _Safe_local_iterator<_Iterator, _Sequence> & __last ) [inline]`

Safe local iterators know how to check if they form a valid range.

Definition at line 147 of file `functions.h`.

3.4.3.14 `template<typename _Integral> bool __gnu_debug::__valid_range_aux ( const _Integral &, const _Integral &, std::__true_type ) [inline]`

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 111 of file `functions.h`.

Referenced by `__valid_range()`.

3.4.3.15 `template<typename _InputIterator> bool __gnu_debug::__valid_range_aux ( const _InputIterator & __first, const _InputIterator & __last, std::__false_type ) [inline]`

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 119 of file `functions.h`.

References `std::iterator_category()`, and `__valid_range_aux2()`.

3.4.3.16 `template<typename _RandomAccessIterator> bool __gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator & __first, const _RandomAccessIterator & __last, std::random_access_iterator_tag ) [inline]`

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 90 of file `functions.h`.

Referenced by `__valid_range_aux()`.

3.4.3.17 `template<typename _InputIterator> bool __gnu_debug::__valid_range_aux2 ( const _InputIterator & , const _InputIterator & , std::input_iterator_tag ) [inline]`

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 101 of file functions.h.

## 3.5 `__gnu_internal` Namespace Reference

### 3.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 3.6 `__gnu_parallel` Namespace Reference

### Classes

- struct [\\_\\_accumulate\\_binop\\_reduct](#)  
*General reduction, using a binary operator.*
- struct [\\_\\_accumulate\\_selector](#)  
*std::accumulate() selector.*
- struct [\\_\\_adjacent\\_difference\\_selector](#)  
*Selector that returns the difference between two adjacent \_\_elements.*
- struct [\\_\\_adjacent\\_find\\_selector](#)  
*Test predicate on two adjacent elements.*
- class [\\_\\_binder1st](#)  
*Similar to std::binder1st, but giving the argument types explicitly.*
- class [\\_\\_binder2nd](#)  
*Similar to std::binder2nd, but giving the argument types explicitly.*
- struct [\\_\\_count\\_if\\_selector](#)  
*std::count\_if() selector.*
- struct [\\_\\_count\\_selector](#)  
*std::count() selector.*
- struct [\\_\\_fill\\_selector](#)  
*std::fill() selector.*
- struct [\\_\\_find\\_first\\_of\\_selector](#)  
*Test predicate on several elements.*
- struct [\\_\\_find\\_if\\_selector](#)  
*Test predicate on a single element, used for std::find() and std::find\_if().*
- struct [\\_\\_for\\_each\\_selector](#)  
*std::for\_each() selector.*
- struct [\\_\\_generate\\_selector](#)  
*std::generate() selector.*
- struct [\\_\\_generic\\_find\\_selector](#)  
*Base class of all \_\_gnu\_parallel::\_\_find\_template selectors.*
- struct [\\_\\_generic\\_for\\_each\\_selector](#)

- Generic `__selector` for embarrassingly parallel functions.
- struct [\\_\\_identity\\_selector](#)

Selector that just returns the passed iterator.
  - struct [\\_\\_inner\\_product\\_selector](#)

`std::inner_product()` selector.
  - struct [\\_\\_max\\_element\\_reduct](#)

Reduction for finding the maximum element, using a comparator.
  - struct [\\_\\_min\\_element\\_reduct](#)

Reduction for finding the maximum element, using a comparator.
  - struct [\\_\\_mismatch\\_selector](#)

Test inverted predicate on a single element.
  - struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch](#)

Switch for 3-way merging with `__sentinels` turned off.
  - struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)

Switch for 3-way merging with `__sentinels` turned on.
  - struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)

Switch for 4-way merging with `__sentinels` turned off.
  - struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)

Switch for 4-way merging with `__sentinels` turned on.
  - struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)

Switch for  $k$ -way merging with `__sentinels` turned on.
  - struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)

Switch for  $k$ -way merging with `__sentinels` turned off.
  - struct [\\_\\_replace\\_if\\_selector](#)

`std::replace()` selector.
  - struct [\\_\\_replace\\_selector](#)

`std::replace()` selector.
  - struct [\\_\\_transform1\\_selector](#)

`std::transform()` `__selector`, one input sequence variant.
  - struct [\\_\\_transform2\\_selector](#)

`std::transform()` `__selector`, two input sequences variant.
  - class [\\_\\_unary\\_negate](#)

Similar to `std::unary_negate`, but giving the argument types explicitly.
  - struct [\\_DRandomShufflingGlobalData](#)

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.
  - struct [\\_DRSSorterPU](#)

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.
  - struct [\\_DummyReduct](#)

Reduction function doing nothing.
  - class [\\_EqualFromLess](#)

Constructs predicate for equality from strict weak ordering predicate.
  - struct [\\_EqualTo](#)

Similar to `std::equal_to`, but allows two different types.
  - class [\\_GuardedIterator](#)

- \_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*
- class [\\_IteratorPair](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_IteratorTriple](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*
- struct [\\_Job](#)  
*One \_\_job for a certain thread.*
- struct [\\_Less](#)  
*Similar to `std::less`, but allows two different types.*
- class [\\_Lexicographic](#)  
*Compare \_\_a pair of types lexicographically, ascending.*
- class [\\_LexicographicReverse](#)  
*Compare \_\_a pair of types lexicographically, descending.*
- class [\\_LoserTree](#)  
*Stable \_LoserTree variant.*
- class [\\_LoserTree< false, \\_Tp, \\_Compare >](#)  
*Unstable \_LoserTree variant.*
- class [\\_LoserTreeBase](#)  
*Guarded loser/tournament tree.*
- class [\\_LoserTreePointer](#)  
*Stable \_LoserTree implementation.*
- class [\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)  
*Unstable \_LoserTree implementation.*
- class [\\_LoserTreePointerBase](#)  
*Base class of \_Loser Tree implementation using pointers.*
- class [\\_LoserTreePointerUnguarded](#)  
*Stable unguarded \_LoserTree variant storing pointers.*
- class [\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)  
*Unstable unguarded \_LoserTree variant storing pointers.*
- class [\\_LoserTreePointerUnguardedBase](#)  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- struct [\\_LoserTreeTraits](#)  
*Traits for determining whether the loser tree should use pointers or copies.*
- class [\\_LoserTreeUnguarded](#)  
*Stable implementation of unguarded \_LoserTree.*
- class [\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)  
*Non-Stable implementation of unguarded \_LoserTree.*
- class [\\_LoserTreeUnguardedBase](#)  
*Base class for unguarded \_LoserTree implementation.*
- struct [\\_Multiplies](#)  
*Similar to `std::multiplies`, but allows two different types.*
- struct [\\_Nothing](#)  
*Functor doing nothing.*
- struct [\\_Piece](#)  
*Subsequence description.*
- struct [\\_Plus](#)  
*Similar to `std::plus`, but allows two different types.*

- struct [\\_PMWMSSortingData](#)  
*Data accessed by all threads.*
- class [\\_PseudoSequence](#)  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class [\\_PseudoSequenceIterator](#)  
*\_Iterator associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.*
- struct [\\_QSBThreadLocal](#)  
*Information local to one thread in the parallel quicksort run.*
- class [\\_RandomNumber](#)  
*Random number generator, based on the Mersenne twister.*
- class [\\_RestrictedBoundedConcurrentQueue](#)  
*Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.*
- struct [\\_SamplingSorter](#)  
*Stable sorting functor.*
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)  
*Non-\_\_stable sorting functor.*
- struct [\\_Settings](#)  
*class \_Settings Run-time settings for the parallel mode including all tunable parameters.*
- struct [\\_SplitConsistently](#)  
*Split consistently.*
- struct [\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split by sampling.*
- struct [\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split by exact splitting.*
- struct [balanced\\_quicksort\\_tag](#)  
*Forces parallel sorting using balanced quicksort at compile time.*
- struct [balanced\\_tag](#)  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct [constant\\_size\\_blocks\\_tag](#)  
*Selects the constant block size variant for `std::find()`.*
- struct [default\\_parallel\\_tag](#)  
*Recommends parallel execution using the default parallel algorithm.*
- struct [equal\\_split\\_tag](#)  
*Selects the equal splitting variant for `std::find()`.*
- struct [exact\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [find\\_tag](#)  
*Base class for `std::find()` variants.*
- struct [growing\\_blocks\\_tag](#)  
*Selects the growing block size variant for `std::find()`.*
- struct [multiway\\_mergesort\\_exact\\_tag](#)  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*
- struct [multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [multiway\\_mergesort\\_tag](#)

*Forces parallel sorting using multiway mergesort at compile time.*

- struct [omp\\_loop\\_static\\_tag](#)

*Recommends parallel execution using OpenMP static load-balancing at compile time.*

- struct [omp\\_loop\\_tag](#)

*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*

- struct [parallel\\_tag](#)

*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*

- struct [quicksort\\_tag](#)

*Forces parallel sorting using unbalanced quicksort at compile time.*

- struct [sampling\\_tag](#)

*Forces parallel merging with exact splitting, at compile time.*

- struct [sequential\\_tag](#)

*Forces sequential execution at compile time.*

- struct [unbalanced\\_tag](#)

*Recommends parallel execution using static load-balancing at compile time.*

### Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

### Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_Parallelism](#) { **sequential**, **parallel\_unbalanced**, **parallel\_balanced**, **parallel\_omp\_loop**, **parallel\_omp\_loop\_static**, **parallel\_taskqueue** }
- enum [\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

### Functions

- template<typename [\\_Tp](#) >  
    [\\_Tp \\_\\_add\\_omp](#) (volatile [\\_Tp](#) \*[\\_\\_ptr](#), [\\_Tp](#) [\\_\\_addend](#))
- template<typename [\\_RAIter](#), typename [\\_DifferenceTp](#) >  
    void [\\_\\_calc\\_borders](#) ([\\_RAIter](#) [\\_\\_elements](#), [\\_DifferenceTp](#) [\\_\\_length](#), [\\_DifferenceTp](#) \*[\\_\\_off](#))
- template<typename [\\_Tp](#) >  
    bool [\\_\\_cas\\_omp](#) (volatile [\\_Tp](#) \*[\\_\\_ptr](#), [\\_Tp](#) [\\_\\_comparand](#), [\\_Tp](#) [\\_\\_replacement](#))
- template<typename [\\_Tp](#) >  
    bool [\\_\\_compare\\_and\\_swap](#) (volatile [\\_Tp](#) \*[\\_\\_ptr](#), [\\_Tp](#) [\\_\\_comparand](#), [\\_Tp](#) [\\_\\_replacement](#))
- template<typename [\\_Iter](#), typename [\\_OutputIterator](#) >  
    [\\_OutputIterator](#) [\\_\\_copy\\_tail](#) (std::pair< [\\_Iter](#), [\\_Iter](#) > [\\_\\_b](#), std::pair< [\\_Iter](#), [\\_Iter](#) > [\\_\\_e](#), [\\_OutputIterator](#) [\\_\\_r](#))
- void [\\_\\_decode2](#) ([\\_CASable](#) [\\_\\_x](#), int &[\\_\\_a](#), int &[\\_\\_b](#))

- `template<typename _RAIter, typename _DifferenceTp >`  
`void __determine_samples ( _PMWSSortingData< _RAIter > *__sd, _DifferenceTp __num_samples)`
- `_CASable __encode2 (int __a, int __b)`
- `template<typename _DifferenceType, typename _OutputIterator >`  
`_OutputIterator __equally_split ( _DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`  
`_DifferenceType __equally_split_point ( _DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __-  
thread_no)`
- `template<typename _Tp >`  
`_Tp __fetch_and_add (volatile _Tp *__ptr, _Tp __addend)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _-  
Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _-  
Pred __pred, _Selector __selector, equal_split_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _-  
Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _-  
Pred __pred, _Selector __selector, constant_size_blocks_tag)`
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result >`  
`_UserOp __for_each_template_random_access ( _Iter __begin, _Iter __end, _UserOp __user_op, _Functionality  
& __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits<  
_Iter >::difference_type __bound, _Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_ed ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_omp_loop ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _-  
Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_omp_loop_static ( _RAIter __begin, _RAIter __end, _Op __o, _Fu  
& __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type  
__bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_workstealing ( _RAIter __begin, _RAIter __end, _Op __op, _Fu & __-  
f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type  
__bound)`
- `_ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const _Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool __is_sorted ( _Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __median_of_three_iterators ( _RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __-  
end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_merge\_advance\_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2`  
`__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __`  
`end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare`  
`__comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __`  
`end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare`  
`__comp)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation`  
`__bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary-`  
`Operation __bin_op, typename std::iterator_traits<_Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary-`  
`Operation __bin_op, typename std::iterator_traits<_Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits<_RAIter >`  
`::difference_type \_\_parallel\_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, \_ThreadIndex __num-`  
`threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=\_Random-`  
`Number())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter >::`  
`difference_type __n, \_ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs\_pu (\_DRSSorterPU<_RAIter, _RandomNumberGenerator > *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator \_\_parallel\_set\_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _`  
`OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_symmetric\_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __`  
`end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _Output-`  
`Iterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_Parallelism __parallelism)`

- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_exact\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default\_parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >`  
`::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_conquer (_QSBThreadLocal< _RAIter > *__tls, _RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __iam, ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >`  
`::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > *__tls, _Compare &__comp, ThreadIndex __iam, bool __wait)`
- `template<typename _RandomNumberGenerator >`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`  
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp >`  
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`_RAIter1 __search_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Pred __pred)`

- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`__RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type>::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator & __rng)`
- `template<typename _Iter>`  
`void __shrink (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length)`
- `template<typename _Iter>`  
`void __shrink_and_double (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice)`
- `void __yield ()`
- `template<typename _Iter, typename _FunctorType>`  
`size_t list_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType & __f, int __oversampling=0)`
- `template<typename _Tp>`  
`const _Tp & max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp>`  
`const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type>::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType & __offset, _Compare __comp=std::less< _Tp >())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>`  
`void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > & __pieces)`

- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-`  
`target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare`  
`>`  
`_RAIter3 multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RA-`  
`Iter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type-`  
`::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _-`  
`RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_-`  
`type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _-`  
`DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _-`  
`DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RA-`  
`IterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _-`  
`RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _-`  
`RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _-`  
`RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _-`  
`RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter,`  
`typename _Compare >`  
`_RAIter3 parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-`  
`target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void parallel\_sort\_mwms\_pu (\_PMWMSortingData< _RAIter > * __sd, _Compare & __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RA-`  
`IterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RA-`  
`IterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RA-`  
`IterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RA-`  
`IterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RA-`  
`IterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs-`  
`_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs-`  
`_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs-`  
`_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs-`  
`_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs-`  
`_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

#### Variables

- `static const int \_CASable\_bits`
- `static const \_CASable \_CASable\_mask`

#### 3.6.1 Detailed Description

GNU parallel code for public use.

#### 3.6.2 Typedef Documentation

##### 3.6.2.1 `typedef unsigned short __gnu_parallel::_BinIndex`

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

##### 3.6.2.2 `typedef int64_t __gnu_parallel::_CASable`

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

##### 3.6.2.3 `typedef uint64_t __gnu_parallel::_SequenceIndex`

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

##### 3.6.2.4 `typedef uint16_t __gnu_parallel::_ThreadIndex`

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

### 3.6.3 Enumeration Type Documentation

#### 3.6.3.1 `enum __gnu_parallel::_AlgorithmStrategy`

Strategies for run-time algorithm selection:

Definition at line 67 of file `types.h`.

#### 3.6.3.2 `enum __gnu_parallel::_FindAlgorithm`

Find algorithms:

Definition at line 106 of file `types.h`.

#### 3.6.3.3 `enum __gnu_parallel::_MultiwayMergeAlgorithm`

Merging algorithms:

Definition at line 85 of file `types.h`.

#### 3.6.3.4 `enum __gnu_parallel::_Parallelism`

Run-time equivalents for the compile-time tags.

Enumerator:

***sequential*** Not parallel.

***parallel\_unbalanced*** Parallel unbalanced (equal-sized chunks).

***parallel\_balanced*** Parallel balanced (work-stealing).

***parallel\_omp\_loop*** Parallel with OpenMP dynamic load-balancing.

***parallel\_omp\_loop\_static*** Parallel with OpenMP static load-balancing.

***parallel\_taskqueue*** Parallel with OpenMP taskqueue construct.

Definition at line 44 of file `types.h`.

#### 3.6.3.5 `enum __gnu_parallel::_PartialSumAlgorithm`

Partial sum algorithms: recursive, linear.

Definition at line 91 of file `types.h`.

#### 3.6.3.6 `enum __gnu_parallel::_SortAlgorithm`

Sorting algorithms:

Definition at line 76 of file `types.h`.

#### 3.6.3.7 `enum __gnu_parallel::_SplittingAlgorithm`

Sorting/merging algorithms: sampling, `__exact`.

Definition at line 98 of file `types.h`.

### 3.6.4 Function Documentation

3.6.4.1 `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::__calc_borders ( _RAIter __elements, _DifferenceTp __length, _DifferenceTp * __off )`

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

#### Parameters

|                         |                                           |
|-------------------------|-------------------------------------------|
| <code>__elements</code> | Begin iterator of sequence to search for. |
| <code>__length</code>   | Length of sequence to search for.         |
| <code>__off</code>      | Returned <code>__offsets</code> .         |

Definition at line 51 of file `search.h`.

Referenced by `__search_template()`.

3.6.4.2 `template<typename _Tp> bool __gnu_parallel::__compare_and_swap ( volatile _Tp * __ptr, _Tp __comparand, _Tp __replacement ) [inline]`

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__ptr</code>         | Pointer to signed integer. |
| <code>__comparand</code>   | Compare value.             |
| <code>__replacement</code> | Replacement value.         |

Definition at line 108 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`, `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, and `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`.

3.6.4.3 `void __gnu_parallel::__decode2 ( _CASable __x, int & __a, int & __b ) [inline]`

Decode two integers from one `gnu_parallel::__CASable`.

#### Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>__x</code> | <code>__gnu_parallel::__CASable</code> to decode integers from.                                                 |
| <code>__a</code> | First integer, to be decoded from the most-significant <code>__CASable_bits/2</code> bits of <code>__x</code> . |
| <code>__b</code> | Second integer, to be encoded in the least-significant <code>__CASable_bits/2</code> bits of <code>__x</code> . |

#### See Also

`__encode2`

Definition at line 133 of file `parallel/base.h`.

References `__CASable_bits`, and `__CASable_mask`.

Referenced by `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`, and `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

3.6.4.4 `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::_determine_samples ( _PMWSSortingData<_RAIter> * __sd, _DifferenceTp __num_samples )`

Select `_M_samples` from a sequence.

#### Parameters

|                            |                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------|
| <code>__sd</code>          | Pointer to algorithm data. Result will be placed in <code>__sd-&gt;_M_samples</code> . |
| <code>__num_samples</code> | Number of <code>_M_samples</code> to select.                                           |

Definition at line 97 of file `multiway_mergesort.h`.

References `__equally_split()`, `__gnu_parallel::_PMWSSortingData<_RAIter>::_M_samples`, `__gnu_parallel::_PMWSSortingData<_RAIter>::_M_source`, and `__gnu_parallel::_PMWSSortingData<_RAIter>::_M_starts`.

3.6.4.5 `_CASable __gnu_parallel::_encode2 ( int __a, int __b ) [inline]`

Encode two integers into one `gnu_parallel::_CASable`.

#### Parameters

|                  |                                                                                           |
|------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code> | First integer, to be encoded in the most-significant <code>_CASable_bits/2</code> bits.   |
| <code>__b</code> | Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits. |

#### Returns

value encoding `__a` and `__b`.

#### See Also

`__decode2`

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::push_front()`.

3.6.4.6 `template<typename _DifferenceType, typename _OutputIterator> _OutputIterator __gnu_parallel::_equally_split ( _DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s )`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0, __n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

#### Parameters

|                            |                    |
|----------------------------|--------------------|
| <code>__n</code>           | Number of elements |
| <code>__num_threads</code> | Number of parts    |
| <code>__s</code>           | Splitters          |

## Returns

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

**3.6.4.7** `template<typename _DifferenceType > _DifferenceType __gnu_parallel::__equally_split_point ( _DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no )`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

## Parameters

|                            |                    |
|----------------------------|--------------------|
| <code>__n</code>           | Number of elements |
| <code>__num_threads</code> | Number of parts    |
| <code>__thread_no</code>   | Number of threads  |

## Returns

splitting point

Definition at line 75 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

**3.6.4.8** `template<typename _Tp > _Tp __gnu_parallel::__fetch_and_add ( volatile _Tp * __ptr, _Tp __addend ) [inline]`

Add a value to a variable, atomically.

## Parameters

|                       |                              |
|-----------------------|------------------------------|
| <code>__ptr</code>    | Pointer to a signed integer. |
| <code>__addend</code> | Value to add.                |

Definition at line 74 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

**3.6.4.9** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1 ,_RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector ) [inline]`

Parallel `std::find`, switch for different algorithms.

## Parameters

|                         |                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                |
| <code>__end1</code>     | End iterator of first sequence.                                                                  |
| <code>__begin2</code>   | Begin iterator of second sequence. Must have same length as first sequence.                      |
| <code>__pred</code>     | Find predicate.                                                                                  |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...) |

**Returns**

Place of finding in both sequences.

Definition at line 60 of file find.h.

References `__gnu_parallel::_Settings::get()`, and `std::make_pair()`.

```
3.6.4.10 template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1,
 _RAIter2> __gnu_parallel::_find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector
 __selector, equal_split_tag)
```

Parallel `std::find`, equal splitting variant.

**Parameters**

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

**Returns**

Place of finding in both sequences.

Definition at line 97 of file find.h.

References `__equally_split()`, and `_GLIBCXX_CALL`.

```
3.6.4.11 template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1,
 _RAIter2> __gnu_parallel::_find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector
 __selector, growing_blocks_tag)
```

Parallel `std::find`, growing block size variant.

**Parameters**

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

**Returns**

Place of finding in both sequences.

**See Also**

`__gnu_parallel::_Settings::find_sequential_search_size`  
`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**3.6.4.12** `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair<_RAIter1, _RAIter2> __gnu_parallel::_find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag )`

Parallel `std::find`, constant block size variant.

#### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

#### Returns

Place of finding in both sequences.

#### See Also

`__gnu_parallel::_Settings::find_sequential_search_size`  
`__gnu_parallel::_Settings::find_block_size` There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**3.6.4.13** `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result> _UserOp __gnu_parallel::_for_each_template_random_access ( _Iter __begin, _Iter __end, _UserOp __user_op, _Functionality & __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits<_Iter>::difference_type __bound, _Parallelism __parallelism_tag )`

Chose the desired algorithm by evaluating `__parallelism_tag`.

#### Parameters

|                                |                                                                                                                                                                 |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>           | Begin iterator of input sequence.                                                                                                                               |
| <code>__end</code>             | End iterator of input sequence.                                                                                                                                 |
| <code>__user_op</code>         | A user-specified functor (comparator, predicate, associative operator,...)                                                                                      |
| <code>__functionality</code>   | functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. <code>accumulate</code> , <code>for_each</code> ,...) |
| <code>__reduction</code>       | Reduction functor.                                                                                                                                              |
| <code>__reduction_start</code> | Initial value for reduction.                                                                                                                                    |
| <code>__output</code>          | Output iterator.                                                                                                                                                |
| <code>__bound</code>           | Maximum number of elements processed.                                                                                                                           |
| <code>__parallelism_tag</code> | Parallelization method                                                                                                                                          |

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

**3.6.4.14** `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_ed ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

#### Parameters

|                       |                                                                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                      |
| <code>__end</code>    | End iterator of element sequence.                                                                                                        |
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, ...)                                                                       |
| <code>__f</code>      | Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                |
| <code>__output</code> | Pointer to position where final result is written to                                                                                     |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                           |

#### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `__equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

**3.6.4.15** `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_omp_loop ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red  
__r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

#### Parameters

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, etc.).                                                                          |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file omp\_loop.h.

Referenced by \_\_for\_each\_template\_random\_access().

```
3.6.4.16 template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f,
_Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

## Parameters

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, ...).                                                                           |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).             |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file omp\_loop\_static.h.

```
3.6.4.17 template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f,
_Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

## Parameters

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__op</code>     | User-supplied functor (comparator, predicate, adding functor, ...).                                                                           |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file workstealing.h.

References `__gnu_debug::__base()`, `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Job<_DifferenceTp>::_M_first`, `__gnu_parallel::_Job<_DifferenceTp>::_M_last`, `__gnu_parallel::_Job<_DifferenceTp>::_M_load`, `__gnu_parallel::_Settings::cache_line_size`, `__gnu_parallel::_Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

**3.6.4.18** `template<typename _Iter, typename _Compare> bool __gnu_parallel::_is_sorted ( _Iter __begin, _Iter __end, _Compare __comp )`

Check whether `[__begin, __end)` is sorted according to `__comp`.

## Parameters

|                      |                             |
|----------------------|-----------------------------|
| <code>__begin</code> | Begin iterator of sequence. |
| <code>__end</code>   | End iterator of sequence.   |
| <code>__comp</code>  | Comparator.                 |

## Returns

`true` if sorted, `false` otherwise.

Definition at line 51 of file checkers.h.

Referenced by `__sequential_multiway_merge()`, `multiway_merge_loser_tree_sentinel()`, and `parallel_multiway_merge()`.

**3.6.4.19** `template<typename _RAIter, typename _Compare> _RAIter __gnu_parallel::_median_of_three_iterators ( _RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp )`

Compute the median of three referenced elements, according to `__comp`.

## Parameters

|                     |                  |
|---------------------|------------------|
| <code>__a</code>    | First iterator.  |
| <code>__b</code>    | Second iterator. |
| <code>__c</code>    | Third iterator.  |
| <code>__comp</code> | Comparator.      |

Definition at line 398 of file parallel/base.h.

Referenced by `__qsb_divide()`.

**3.6.4.20** `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp ) [inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

## Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

## Returns

Output end iterator.

Definition at line 171 of file `merge.h`.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

```
3.6.4.21 template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename
 _Compare > _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 &
 __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

## Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

## Returns

Output end iterator.

Definition at line 105 of file `merge.h`.

Referenced by `__merge_advance()`.

```
3.6.4.22 template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename
 _Compare > _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 &
 __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

## Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

## Returns

Output end iterator.

Definition at line 57 of file `merge.h`.

```
3.6.4.23 template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare > _RAIter3
 __gnu_parallel::__parallel_merge_advance (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2,
 _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
 [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

## Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

## Returns

Output end iterator.

Definition at line 195 of file `merge.h`.

References `__merge_advance()`.

```
3.6.4.24 template<typename _RAIter1 , typename _RAIter3 , typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge-
advance (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1 __end2, _RAIter3 __target, typename
std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp) [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

## Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__end2</code>   | End iterator of second sequence.   |

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

Definition at line 223 of file `merge.h`.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**3.6.4.25** `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::nth_element()`.

**Parameters**

|                      |                                                          |
|----------------------|----------------------------------------------------------|
| <code>__begin</code> | Begin iterator of input sequence.                        |
| <code>__nth</code>   | Iterator of element that must be in position afterwards. |
| <code>__end</code>   | End iterator of input sequence.                          |
| <code>__comp</code>  | Comparator.                                              |

Definition at line 332 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::Settings::get()`, `std::max()`, `__gnu_parallel::Settings::nth_element_minimal_n`, and `__gnu_parallel::Settings::partition_minimal_n`.

Referenced by `__parallel_partial_sort()`.

**3.6.4.26** `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::partial_sort()`.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence. |
| <code>__middle</code> | Sort until this position.         |
| <code>__end</code>    | End iterator of input sequence.   |
| <code>__comp</code>   | Comparator.                       |

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

**3.6.4.27** `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator __gnu_parallel::__parallel_partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op )`

Parallel partial sum front-`__end`.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__result</code> | Begin iterator of output sequence. |
| <code>__bin_op</code> | Associative binary function.       |

**Returns**

End iterator of output sequence.

Definition at line 205 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

**3.6.4.28** `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator  
__gnu_parallel::_parallel_partial_sum_basecase ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__bin_op, typename std::iterator_traits< _Iter >::value_type __value )`

Base case prefix sum routine.

**Parameters**

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence.                                            |
| <code>__end</code>    | End iterator of input sequence.                                              |
| <code>__result</code> | Begin iterator of output sequence.                                           |
| <code>__bin_op</code> | Associative binary function.                                                 |
| <code>__value</code>  | Start value. Must be passed since the neutral element is unknown in general. |

**Returns**

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

**3.6.4.29** `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator  
__gnu_parallel::_parallel_partial_sum_linear ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__bin_op, typename std::iterator_traits< _Iter >::difference_type __n )`

Parallel partial sum implementation, two-phase approach, no recursion.

**Parameters**

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence.  |
| <code>__end</code>    | End iterator of input sequence.    |
| <code>__result</code> | Begin iterator of output sequence. |
| <code>__bin_op</code> | Associative binary function.       |
| <code>__n</code>      | Length of sequence.                |

**Returns**

End iterator of output sequence.

Definition at line 89 of file `partial_sum.h`.

References `__equally_split()`, `__parallel_partial_sum_basecase()`, `std::accumulate()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

3.6.4.30 `template<typename _RAIter, typename _Predicate> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__parallel_partition ( _RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads )`

Parallel implementation of `std::partition`.

#### Parameters

|                            |                                                             |
|----------------------------|-------------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of input sequence to split.                  |
| <code>__end</code>         | End iterator of input sequence to split.                    |
| <code>__pred</code>        | Partition predicate, possibly including some kind of pivot. |
| <code>__num_threads</code> | Maximum number of threads to use for this task.             |

#### Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file `partition.h`.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::partition_chunk_share`, and `__gnu_parallel::Settings::partition_chunk_size`.

Referenced by `__parallel_nth_element()`, `__parallel_sort_qs_divide()`, and `__qsb_divide()`.

3.6.4.31 `template<typename _RAIter, typename _RandomNumberGenerator> void __gnu_parallel::__parallel_random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng = _RandomNumber() ) [inline]`

Parallel random public call.

#### Parameters

|                      |                                 |
|----------------------|---------------------------------|
| <code>__begin</code> | Begin iterator of sequence.     |
| <code>__end</code>   | End iterator of sequence.       |
| <code>__rng</code>   | Random number generator to use. |

Definition at line 522 of file `random_shuffle.h`.

References `__parallel_random_shuffle_drs()`.

3.6.4.32 `template<typename _RAIter, typename _RandomNumberGenerator> void __gnu_parallel::__parallel_random_shuffle_drs ( _RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator & __rng )`

Main parallel random shuffle step.

#### Parameters

|                            |                                 |
|----------------------------|---------------------------------|
| <code>__begin</code>       | Begin iterator of sequence.     |
| <code>__end</code>         | End iterator of sequence.       |
| <code>__n</code>           | Length of sequence.             |
| <code>__num_threads</code> | Number of threads to use.       |
| <code>__rng</code>         | Random number generator to use. |

Definition at line 265 of file `random_shuffle.h`.

References \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_bins\_end, \_\_parallel\_random\_shuffle\_drs\_pu(), \_\_rd\_log2(), \_\_round\_up\_to\_pow2(), \_\_sequential\_random\_shuffle(), \_GLIBCXX\_CALL, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_bin\_proc, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_bins\_begin, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_dist, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_num\_bins, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_num\_bits, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_num\_threads, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_sd, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_seed, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_starts, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_temporaries, \_\_gnu\_parallel::Settings::get(), \_\_gnu\_parallel::Settings::L2\_cache\_size, std::min(), and \_\_gnu\_parallel::Settings::TLB\_size.

Referenced by \_\_parallel\_random\_shuffle().

**3.6.4.33** `template<typename _RAIter, typename _RandomNumberGenerator> void __gnu_parallel::__parallel_random_shuffle_drs_pu ( _DRSSorterPU<_RAIter, _RandomNumberGenerator> * __pus )`

Random shuffle code executed by each thread.

#### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pus</code> | Array of thread-local data records. |
|--------------------|-------------------------------------|

Definition at line 122 of file random\_shuffle.h.

References \_\_random\_number\_pow2(), \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_dist, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_num\_bins, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_num\_bits, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_num\_threads, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_sd, \_\_gnu\_parallel::DRSSorterPU<\_RAIter, \_RandomNumberGenerator>::\_\_M\_seed, \_\_gnu\_parallel::DRandomShufflingGlobalData<\_RAIter>::\_\_M\_starts, and std::partial\_sum().

Referenced by \_\_parallel\_random\_shuffle\_drs().

**3.6.4.34** `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism ) [inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

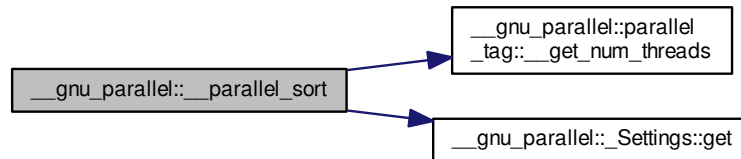
#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 75 of file sort.h.

References \_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads(), \_GLIBCXX\_CALL, and \_\_gnu\_parallel::Settings::get().

Here is the call graph for this function:



3.6.4.35 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

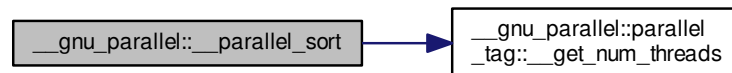
#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 99 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.36 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism ) [inline]`

Choose multiway mergesort with splitting by sampling, for parallel sorting.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

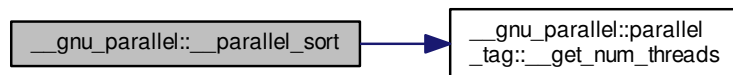
## Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 120 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**3.6.4.37** `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism ) [inline]`

Choose quicksort for parallel sorting.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

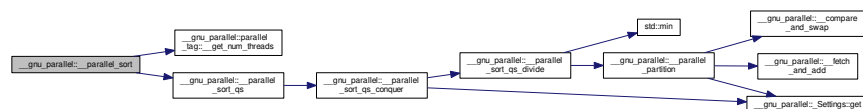
## Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 140 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.38 `template<bool __stable, typename _RAIter, typename _Compare > void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism ) [inline]`

Choose balanced quicksort for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

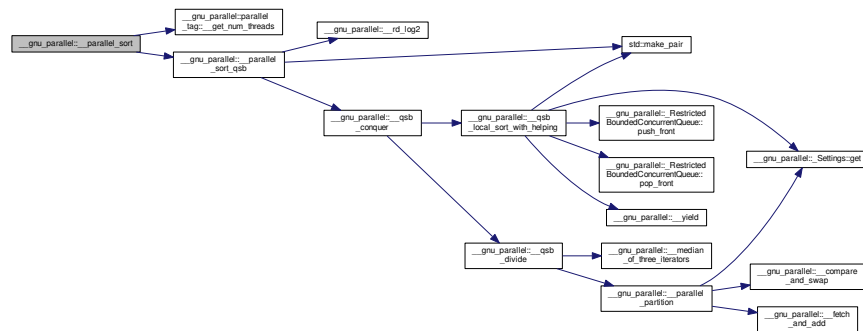
#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 161 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.39 `template<bool __stable, typename _RAIter, typename _Compare > void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

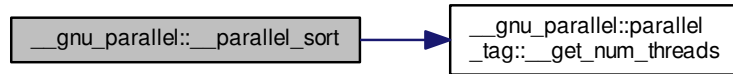
#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 183 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.40 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism ) [inline]`

Choose a parallel sorting algorithm.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

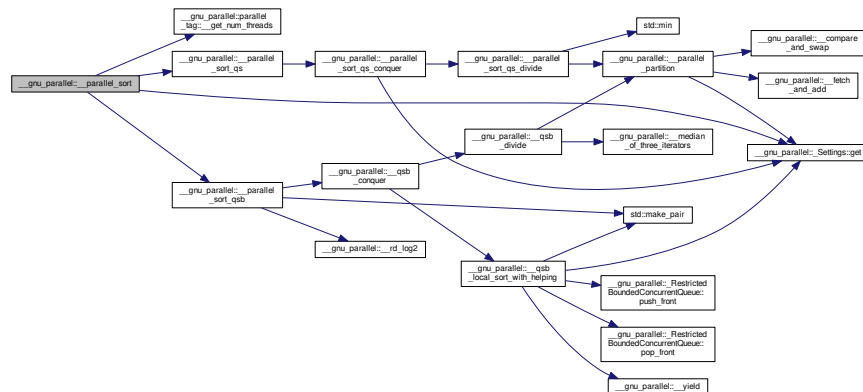
#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 203 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::Settings::get()`.

Here is the call graph for this function:



3.6.4.41 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qs ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort main call.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of input sequence.                        |
| <code>__end</code>         | End iterator input sequence, ignored.                    |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 154 of file `quicksort.h`.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

3.6.4.42 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qs_conquer ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort conquer step.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 101 of file `quicksort.h`.

References `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`.

3.6.4.43 `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_sort_qs_divide ( _RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits<_RAIter>::difference_type __pivot_rank, typename std::iterator_traits<_RAIter>::difference_type __num_samples, _ThreadIndex __num_threads )`

Unbalanced quicksort divide step.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__pivot_rank</code>  | Desired <code>__rank</code> of the pivot.                |
| <code>__num_samples</code> | Choose pivot from that many samples.                     |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 51 of file `quicksort.h`.

References `__parallel_partition()`, and `std::min()`.

Referenced by `__parallel_sort_qs_conquer()`.

3.6.4.44 `template<typename _RAIter, typename _Compare > void __gnu_parallel::__parallel_sort_qsb ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Top-level quicksort routine.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of sequence.                              |
| <code>__end</code>         | End iterator of sequence.                                |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 430 of file `balanced_quicksort.h`.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, `__gnu_parallel::QSBThreadLocal< _RAIter >::M_elements_leftover`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

3.6.4.45 `template<typename _Iter, class _OutputIterator, class _BinaryPredicate > _OutputIterator __gnu_parallel::__parallel_unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred )`

Parallel `std::unique_copy()`, w/ `__o` explicit equality predicate.

#### Parameters

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <code>__first</code>       | Begin iterator of input sequence.                  |
| <code>__last</code>        | End iterator of input sequence.                    |
| <code>__result</code>      | Begin iterator of result <code>__sequence</code> . |
| <code>__binary_pred</code> | Equality predicate.                                |

#### Returns

End iterator of result `__sequence`.

Definition at line 50 of file `unique_copy.h`.

References `__equally_split()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_unique_copy()`.

3.6.4.46 `template<typename _Iter, class _OutputIterator > _OutputIterator __gnu_parallel::__parallel_unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result ) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.

#### Parameters

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <code>__first</code>  | Begin iterator of input sequence.                  |
| <code>__last</code>   | End iterator of input sequence.                    |
| <code>__result</code> | Begin iterator of result <code>__sequence</code> . |

#### Returns

End iterator of result `__sequence`.

Definition at line 186 of file `unique_copy.h`.

References \_\_parallel\_unique\_copy().

3.6.4.47 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__qsb_conquer ( _QSBThreadLocal< _RAIter > ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait )`

Quicksort conquer step.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__tls</code>         | Array of thread-local storages.                          |
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__iam</code>         | Number of the thread processing this function.           |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 171 of file `balanced_quicksort.h`.

References \_\_qsb\_divide(), \_\_qsb\_local\_sort\_with\_helping(), \_\_gnu\_parallel::\_\_QSBThreadLocal< \_RAIter >::\_\_M\_elements\_leftover, and \_\_gnu\_parallel::\_\_QSBThreadLocal< \_RAIter >::\_\_M\_initial.

Referenced by \_\_parallel\_sort\_qsb().

3.6.4.48 `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Balanced quicksort divide step.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

#### Precondition

`(__end-__begin)>=1`

Definition at line 100 of file `balanced_quicksort.h`.

References \_\_median\_of\_three\_iterators(), and \_\_parallel\_partition().

Referenced by \_\_qsb\_conquer().

3.6.4.49 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__qsb_local_sort_with_helping ( _QSBThreadLocal< _RAIter > ** __tls, _Compare & __comp, _ThreadIndex __iam, bool __wait )`

Quicksort step doing load-balanced local sort.

#### Parameters

|                     |                                                |
|---------------------|------------------------------------------------|
| <code>__tls</code>  | Array of thread-local storages.                |
| <code>__comp</code> | Comparator.                                    |
| <code>__iam</code>  | Number of the thread processing this function. |

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_elements_leftover`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_initial`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_leftover_parts`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_num_threads`, `__gnu_parallel::__Settings::get()`, `std::make_pair()`, `__gnu_parallel::__RestrictedBoundedConcurrentQueue<_Tp>::pop_front()`, `__gnu_parallel::__RestrictedBoundedConcurrentQueue<_Tp>::push_front()`, and `__gnu_parallel::__Settings::sort_qsb_base_case_maximal_n`.

Referenced by `__qsb_conquer()`.

**3.6.4.50** `template<typename _RandomNumberGenerator> int __gnu_parallel::__random_number_pow2 ( int __logp, _RandomNumberGenerator & __rng ) [inline]`

Generate a random number in  $[0, 2^{\text{__logp}})$ .

#### Parameters

|                     |                                                               |
|---------------------|---------------------------------------------------------------|
| <code>__logp</code> | Logarithm (basis 2) of the upper range <code>__bound</code> . |
| <code>__rng</code>  | Random number generator to use.                               |

Definition at line 115 of file `random_shuffle.h`.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

**3.6.4.51** `template<typename _Size> _Size __gnu_parallel::__rd_log2 ( _Size __n ) [inline]`

Calculates the rounded-down logarithm of `__n` for base 2.

#### Parameters

|                  |           |
|------------------|-----------|
| <code>__n</code> | Argument. |
|------------------|-----------|

#### Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file `parallel/base.h`.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::__LoserTreeBase<_Tp, _Compare>::__LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

**3.6.4.52** `template<typename _Tp> _Tp __gnu_parallel::__round_up_to_pow2 ( _Tp __x )`

Round up to the next greater power of 2.

#### Parameters

|                  |                                   |
|------------------|-----------------------------------|
| <code>__x</code> | <code>_Integer</code> to round up |
|------------------|-----------------------------------|

Definition at line 248 of file `random_shuffle.h`.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

3.6.4.53 `template<typename _RAIter1, typename _RAIter2, typename _Pred> _RAIter1 __gnu_parallel::__search_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Pred __pred )`

Parallel std::search.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__end2</code>   | End iterator of second sequence.   |
| <code>__pred</code>   | Find predicate.                    |

#### Returns

Place of finding in first sequences.

Definition at line 81 of file search.h.

References `__calc_borders()`, `__equally_split()`, `_GLIBCXX_CALL`, and `std::min()`.

3.6.4.54 `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 __gnu_parallel::__sequential_multiway_merge ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type>::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )`

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

#### Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                 |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                   |
| <code>__target</code>     | Begin iterator of output sequence.                                              |
| <code>__comp</code>       | Comparator.                                                                     |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |
| <code>__sentinel</code>   | The sequences have <code>__a</code> <code>__sentinel</code> element.            |

#### Returns

End iterator of output sequence.

Definition at line 920 of file multiway\_merge.h.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

3.6.4.55 `template<typename _RAIter, typename _RandomNumberGenerator> void __gnu_parallel::__sequential_random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator & __rng )`

Sequential cache-efficient random shuffle.

#### Parameters

|                      |                                 |
|----------------------|---------------------------------|
| <code>__begin</code> | Begin iterator of sequence.     |
| <code>__end</code>   | End iterator of sequence.       |
| <code>__rng</code>   | Random number generator to use. |

Definition at line 410 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, `std::partial_sum()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**3.6.4.56** `template<typename _Iter > void __gnu_parallel::_shrink ( std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length )`

Combines two ranges into one and thus halves the number of ranges.

#### Parameters

|                             |                                          |
|-----------------------------|------------------------------------------|
| <code>__os_starts</code>    | Start positions worked on (oversampled). |
| <code>__count_to_two</code> | Counts up to 2.                          |
| <code>__range_length</code> | Current length of a chunk.               |

Definition at line 70 of file `list_partition.h`.

References `std::vector< _Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

**3.6.4.57** `template<typename _Iter > void __gnu_parallel::_shrink_and_double ( std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice )`

Shrinks and doubles the ranges.

#### Parameters

|                             |                                                                    |
|-----------------------------|--------------------------------------------------------------------|
| <code>__os_starts</code>    | Start positions worked on (oversampled).                           |
| <code>__count_to_two</code> | Counts up to 2.                                                    |
| <code>__range_length</code> | Current length of a chunk.                                         |
| <code>__make_twice</code>   | Whether the <code>__os_starts</code> is allowed to be grown or not |

Definition at line 50 of file `list_partition.h`.

References `__shrink()`, `std::vector< _Tp, _Alloc >::resize()`, and `std::vector< _Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

**3.6.4.58** `void __gnu_parallel::_yield ( ) [inline]`

Yield control to another thread, without waiting for the end of the time slice.

Definition at line 121 of file `parallel/compatibility.h`.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

**3.6.4.59** `template<typename _Iter, typename _FuncType > size_t __gnu_parallel::list_partition ( const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FuncType & __f, int __oversampling = 0 )`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |

|                             |                                                                                                                                                                                                                                                           |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__starts</code>       | Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts[ __num_parts]</code> contains the end iterator of the sequence.                                                                           |
| <code>__lengths</code>      | Length of the resulting parts.                                                                                                                                                                                                                            |
| <code>__num_parts</code>    | Number of parts to split the sequence into.                                                                                                                                                                                                               |
| <code>__f</code>            | Functor to be applied to each element by traversing <code>__it</code>                                                                                                                                                                                     |
| <code>__oversampling</code> | Oversampling factor. If 0, then the partitions will differ in at most $\{ \{ \_end \} - \{ \_begin \} \}$ <code>__elements</code> . Otherwise, the ratio between the longest and the shortest part is bounded by $1 / ( \{ \_oversampling \} \{ num \} )$ |

**Returns**

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector< _Tp, _Alloc >::size()`.

**3.6.4.60** `template<typename _Tp> const _Tp& __gnu_parallel::max( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

**3.6.4.61** `template<typename _Tp> const _Tp& __gnu_parallel::min( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`.

**3.6.4.62** `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare> void __gnu_parallel::multiseq_partition( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp = std::less< typename std::iterator_traits<typename std::iterator_traits<_RanSeqs>::value_type::first_type>::value_type>() )`

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

**Parameters**

|                              |                                                                                                                                                                                                                                                              |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin_seqs</code>    | Begin of the sequence of iterator pairs.                                                                                                                                                                                                                     |
| <code>__end_seqs</code>      | End of the sequence of iterator pairs.                                                                                                                                                                                                                       |
| <code>__rank</code>          | The global rank to partition at.                                                                                                                                                                                                                             |
| <code>__begin_offsets</code> | A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> . |
| <code>__comp</code>          | The ordering functor, defaults to <code>std::less&lt;_Tp&gt;</code> .                                                                                                                                                                                        |

Definition at line 122 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector< _Tp, _Alloc >::begin()`, `std::distance()`, `std::priority_queue< _Tp, _Sequence, _Compare >::empty()`, `std::vector< _Tp, _Alloc >::end()`, `std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`, `std::priority_queue< _Tp, _Sequence, _Compare >::push()`,

`std::vector< _Tp, _Alloc >::push_back()`, and `std::priority_queue< _Tp, _Sequence, _Compare >::top()`.

Referenced by `multiway_merge_exact_splitting()`.

**3.6.4.63** `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare > _Tp  
__gnu_parallel::multiseq_selection ( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &  
__offset, _Compare __comp = std::less<_Tp> ( ) )`

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

#### Parameters

|                           |                                                                                                                                                            |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin_seqs</code> | Begin of the sequence of iterator pairs.                                                                                                                   |
| <code>__end_seqs</code>   | End of the sequence of iterator pairs.                                                                                                                     |
| <code>__rank</code>       | The global rank to partition at.                                                                                                                           |
| <code>__offset</code>     | The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0. |
| <code>__comp</code>       | The ordering functor, defaults to <code>std::less</code> .                                                                                                 |

Definition at line 388 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector< _Tp, _Alloc >::begin()`, `std::distance()`, `std::priority_queue< _Tp, _Sequence, _Compare >::empty()`, `std::vector< _Tp, _Alloc >::end()`, `std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`, `std::priority_queue< _Tp, _Sequence, _Compare >::push()`, `std::vector< _Tp, _Alloc >::push_back()`, and `std::priority_queue< _Tp, _Sequence, _Compare >::top()`.

**3.6.4.64** `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut __gnu_parallel::multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 10; ++__j)
 sequences[__i][__j] = __j;
```

```
int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int*>(), 33);
```

**See Also**

`stable_multiway_merge`

**Precondition**

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

**Postcondition**

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
 return `__value` - `__target` = min(`__length`, number of elements in all sequences).

**Template Parameters**

|                                   |                                                            |
|-----------------------------------|------------------------------------------------------------|
| <code>__RAIterPairIterator</code> | iterator over sequence of pairs of iterators               |
| <code>__RAIterOut</code>          | iterator over target sequence                              |
| <code>__DifferenceTp</code>       | difference type for the sequence                           |
| <code>__Compare</code>            | strict weak ordering type to compare elements in sequences |

**Parameters**

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | __begin of sequence <code>__sequence</code>                                     |
| <code>__seqs_end</code>   | __M_end of sequence <code>__sequence</code>                                     |
| <code>__target</code>     | target sequence to merge to.                                                    |
| <code>__comp</code>       | strict weak ordering to use for element comparison.                             |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

**Returns**

`__M_end` iterator of output sequence

Definition at line 1418 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `__GLIBCXX_CALL`.

**3.6.4.65** `template<template< typename RAI, typename C > class iterator, typename __RAIterIterator , typename __RAIter3 , typename __DifferenceTp , typename __Compare > __RAIter3 __gnu_parallel::multiway_merge_3_variant ( __RAIterIterator __seqs_begin, __RAIterIterator __seqs_end, __RAIter3 __target, __DifferenceTp __length, __Compare __comp )`

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the

number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

#### Returns

End iterator of output sequence.

Definition at line 241 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**3.6.4.66** `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_4_variant ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp )`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 360 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

```
3.6.4.67 template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType > void
 __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType
 __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType
 > > * __pieces)
```

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1120 of file `multiway_merge.h`.

References `__equally_split()`, `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`.

```
3.6.4.68 template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >
 _RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3
 __target, _DifferenceTp __length, _Compare __comp)
```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

**Parameters**

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 491 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

```
3.6.4.69 template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
 typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits<
 _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)
```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

## Template Parameters

|                           |                                                                    |
|---------------------------|--------------------------------------------------------------------|
| <i>UnguardedLoserTree</i> | <code>_Loser</code> Tree variant to use for the unguarded merging. |
|---------------------------|--------------------------------------------------------------------|

## Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

## Returns

End iterator of output sequence.

Definition at line 662 of file `multiway_merge.h`.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

```
3.6.4.70 template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare
> _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator
__seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)
```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the `LoserTree` class `_LT`.

Stability is selected by the used `LoserTrees`.

## Precondition

No input will run out of elements during the merge.

## Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

## Returns

End iterator of output sequence.

Definition at line 574 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

```
3.6.4.71 template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType > void
__gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,
_DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType,
_DifferenceType > > * __pieces)
```

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1035 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

```
3.6.4.72 template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
 _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,
 _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 11; ++__j)
 sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

## Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
 return `__value` - `__target` = min(`__length`, number of elements in all sequences).

## See Also

`stable_multiway_merge_sentinels`

## Template Parameters

|                                  |                                                            |
|----------------------------------|------------------------------------------------------------|
| <code>_RAIterPairIterator</code> | iterator over sequence of pairs of iterators               |
| <code>_RAIterOut</code>          | iterator over target sequence                              |
| <code>_DifferenceTp</code>       | difference type for the sequence                           |
| <code>_Compare</code>            | strict weak ordering type to compare elements in sequences |

## Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | <code>__begin</code> of sequence <code>__sequence</code>                        |
| <code>__seqs_end</code>   | <code>_M_end</code> of sequence <code>__sequence</code>                         |
| <code>__target</code>     | target sequence to merge to.                                                    |
| <code>__comp</code>       | strict weak ordering to use for element comparison.                             |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

## Returns

`_M_end` iterator of output sequence

Definition at line 1782 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

3.6.4.73 `template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp ,  
 typename _Splitter , typename _Compare > _RAIter3 __gnu_parallel::parallel_multiway_merge ( _RAIterIterator __seqs_begin,  
 _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex  
 __num_threads )`

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

## Template Parameters

|                         |                                                                        |
|-------------------------|------------------------------------------------------------------------|
| <code>_Splitter</code>  | functor to split input (either <code>__exact</code> or sampling based) |
| <code>__stable</code>   | Stable merging incurs a performance penalty.                           |
| <code>__sentinel</code> | Ignored.                                                               |

## Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                 |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                   |
| <code>__target</code>     | Begin iterator of output sequence.                                              |
| <code>__comp</code>       | Comparator.                                                                     |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

## Returns

End iterator of output sequence.

Definition at line 1225 of file `multiway_merge.h`.

References `__is_sorted()`, `_GLIBCXX_CALL`, `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, and `__gnu_parallel::_Settings::merge_oversampling`.

Referenced by `__parallel_merge_advance()`.

**3.6.4.74** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void __gnu_parallel::parallel_sort_mwms ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

PMWMS main call.

## Parameters

|                            |                             |
|----------------------------|-----------------------------|
| <code>__begin</code>       | Begin iterator of sequence. |
| <code>__end</code>         | End iterator of sequence.   |
| <code>__comp</code>        | Comparator.                 |
| <code>__num_threads</code> | Number of threads to use.   |

Definition at line 395 of file `multiway_mergesort.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_num_threads`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_offsets`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_pieces`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_samples`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_source`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_starts`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_temporary`, `__gnu_parallel::_Settings::get()`, `std::vector<_Tp, _Alloc>::resize()`, and `__gnu_parallel::_Settings::sort_mwms_oversampling`.

**3.6.4.75** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void __gnu_parallel::parallel_sort_mwms_pu ( _PMWMSortingData<_RAIter> * __sd, _Compare & __comp )`

PMWMS code executed by each thread.

## Parameters

|                     |                            |
|---------------------|----------------------------|
| <code>__sd</code>   | Pointer to algorithm data. |
| <code>__comp</code> | Comparator.                |

Definition at line 308 of file `multiway_mergesort.h`.

References `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_num_threads`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_pieces`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_source`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_starts`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_temporary`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, `__gnu_parallel::_Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

## 3.6.5 Variable Documentation

**3.6.5.1** `const int __gnu_parallel::_CASable_bits` `[static]`

Number of bits of `_CASable`.

Definition at line 130 of file `types.h`.

Referenced by `__decode2()`, and `__encode2()`.

#### 3.6.5.2 `const _CASable __gnu_parallel::_CASable_mask` `[static]`

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file `types.h`.

Referenced by `__decode2()`.

## 3.7 \_\_gnu\_pbds Namespace Reference

### Classes

- struct [associative\\_tag](#)  
*Basic associative-container.*
- class [basic\\_branch](#)
- struct [basic\\_branch\\_tag](#)  
*Basic branch structure.*
- class [basic\\_hash\\_table](#)
- struct [basic\\_hash\\_tag](#)  
*Basic hash structure.*
- struct [basic\\_invalidation\\_guarantee](#)
- struct [binary\\_heap\\_tag](#)  
*Binary-heap (array-based).*
- struct [binomial\\_heap\\_tag](#)  
*Binomial-heap.*
- class [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)  
*A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.*
- class [cc\\_hash\\_table](#)
- struct [cc\\_hash\\_tag](#)  
*Collision-chaining hash.*
- struct [container\\_error](#)  
*Base class for exceptions.*
- struct [container\\_tag](#)  
*Base data structure tag.*
- struct [container\\_traits](#)  
*Container traits.*
- struct [container\\_traits\\_base< binary\\_heap\\_tag >](#)  
*Specialization, binary heap.*
- struct [container\\_traits\\_base< binomial\\_heap\\_tag >](#)  
*Specialization, binomial heap.*
- struct [container\\_traits\\_base< cc\\_hash\\_tag >](#)  
*Specialization, cc hash.*
- struct [container\\_traits\\_base< gp\\_hash\\_tag >](#)  
*Specialization, gp hash.*
- struct [container\\_traits\\_base< list\\_update\\_tag >](#)  
*Specialization, list update.*
- struct [container\\_traits\\_base< ov\\_tree\\_tag >](#)

- Specialization, ov tree.*
- struct `container_traits_base< pairing_heap_tag >`
  - Specialization, pairing heap.*
- struct `container_traits_base< pat_trie_tag >`
  - Specialization, pat trie.*
- struct `container_traits_base< rb_tree_tag >`
  - Specialization, rb tree.*
- struct `container_traits_base< rc_binomial_heap_tag >`
  - Specialization, rc binomial heap.*
- struct `container_traits_base< splay_tree_tag >`
  - Specialization, splay tree.*
- struct `container_traits_base< thin_heap_tag >`
  - Specialization, thin heap.*
- class `direct_mask_range_hashing`
  - A mask range-hashing class (uses a bitmask).*
- class `direct_mod_range_hashing`
  - A mod range-hashing class (uses the modulo function).*
- class `gp_hash_table`
- struct `gp_hash_tag`
  - General-probing hash.*
- class `hash_exponential_size_policy`
  - A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).*
- class `hash_load_check_resize_trigger`
  - A resize trigger policy based on a load check. It keeps the load factor between some load factors `load_min` and `load_max`.*
- class `hash_prime_size_policy`
  - A size policy whose sequence of sizes form a nearly-exponential sequence of primes.*
- class `hash_standard_resize_policy`
  - A resize policy which delegates operations to size and trigger policies.*
- struct `insert_error`
  - An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).*
- struct `join_error`
  - A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.*
- class `linear_probe_fn`
  - A probe sequence policy using fixed increments.*
- class `list_update`
- struct `list_update_tag`
  - List-update.*
- class `lu_counter_policy`
- class `lu_move_to_front_policy`
- struct `null_node_update`
  - A null node updatator, indicating that no node updates are required.*
- struct `null_type`
  - Represents no type, or absence of type, for template tricks.*
- struct `ov_tree_tag`
  - Ordered-vector tree.*
- struct `pairing_heap_tag`

- Pairing-heap.*
- struct [pat\\_trie\\_tag](#)
  - PATRICIA trie.*
- struct [point\\_invalidation\\_guarantee](#)
- class [priority\\_queue](#)
- struct [priority\\_queue\\_tag](#)
  - Basic priority-queue.*
- class [quadratic\\_probe\\_fn](#)
  - A probe sequence policy using square increments.*
- struct [range\\_invalidation\\_guarantee](#)
- struct [rb\\_tree\\_tag](#)
  - Red-black tree.*
- struct [rc\\_binomial\\_heap\\_tag](#)
  - Redundant-counter binomial-heap.*
- struct [resize\\_error](#)
  - A container cannot be resized.*
- class [sample\\_probe\\_fn](#)
  - A sample probe policy.*
- class [sample\\_range\\_hashing](#)
  - A sample range-hashing functor.*
- class [sample\\_ranged\\_hash\\_fn](#)
  - A sample ranged-hash functor.*
- class [sample\\_ranged\\_probe\\_fn](#)
  - A sample ranged-probe functor.*
- class [sample\\_resize\\_policy](#)
  - A sample resize policy.*
- class [sample\\_resize\\_trigger](#)
  - A sample resize trigger policy.*
- class [sample\\_size\\_policy](#)
  - A sample size policy.*
- class [sample\\_tree\\_node\\_update](#)
  - A sample node updatator.*
- struct [sample\\_trie\\_access\\_traits](#)
  - A sample trie element access traits.*
- class [sample\\_trie\\_node\\_update](#)
  - A sample node updatator.*
- struct [sample\\_update\\_policy](#)
  - A sample list-update policy.*
- struct [sequence\\_tag](#)
  - Basic sequence.*
- struct [splay\\_tree\\_tag](#)
  - Splay tree.*
- struct [string\\_tag](#)
  - Basic string container, inclusive of strings, ropes, etc.*
- struct [thin\\_heap\\_tag](#)
  - Thin heap.*
- class [tree](#)

- class [tree\\_order\\_statistics\\_node\\_update](#)  
*Functor updating ranks of entrees.*
- struct [tree\\_tag](#)  
*Basic tree structure.*
- class [trie](#)
- class [trie\\_order\\_statistics\\_node\\_update](#)  
*Functor updating ranks of entrees.*
- class [trie\\_prefix\\_search\\_node\\_update](#)  
*A node upator that allows tries to be searched for the range of values that match a certain prefix.*
- struct [trie\\_string\\_access\\_traits](#)
- struct [trie\\_tag](#)  
*Basic trie structure.*
- struct [trivial\\_iterator\\_tag](#)  
*A trivial iterator tag. Signifies that the iterators has none of `std::iterators`'s movement abilities.*

#### Typedefs

- typedef void [trivial\\_iterator\\_difference\\_type](#)

#### Functions

- void [\\_\\_throw\\_container\\_error](#) ()
- void [\\_\\_throw\\_insert\\_error](#) ()
- void [\\_\\_throw\\_join\\_error](#) ()
- void [\\_\\_throw\\_resize\\_error](#) ()

#### 3.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 3.8 `__gnu_profile` Namespace Reference

#### Classes

- class [\\_\\_container\\_size\\_info](#)  
*A container size instrumentation line in the object table.*
- class [\\_\\_container\\_size\\_stack\\_info](#)  
*A container size instrumentation line in the stack table.*
- class [\\_\\_hashfunc\\_info](#)  
*A hash performance instrumentation line in the object table.*
- class [\\_\\_hashfunc\\_stack\\_info](#)  
*A hash performance instrumentation line in the stack table.*
- class [\\_\\_list2vector\\_info](#)  
*A list-to-vector instrumentation line in the object table.*
- class [\\_\\_map2umap\\_info](#)  
*A map-to-unordered\_map instrumentation line in the object table.*
- class [\\_\\_map2umap\\_stack\\_info](#)

- A map-to-unordered\_map instrumentation line in the stack table.*
- class `__object_info_base`  
*Base class for a line in the object table.*
- struct `__reentrance_guard`  
*Reentrance guard.*
- class `__stack_hash`  
*Hash function for summary trace using call stack as index.*
- class `__stack_info_base`  
*Base class for a line in the stack table.*
- class `__trace_base`  
*Base class for all trace producers.*
- class `__trace_container_size`  
*Container size instrumentation trace producer.*
- class `__trace_hash_func`  
*Hash performance instrumentation producer.*
- class `__trace_hashtable_size`  
*Hashtable size instrumentation trace producer.*
- class `__trace_map2umap`  
*Map-to-unordered\_map instrumentation producer.*
- class `__trace_vector_size`  
*Hashtable size instrumentation trace producer.*
- class `__trace_vector_to_list`  
*Vector-to-list instrumentation producer.*
- class `__vector2list_info`  
*A vector-to-list instrumentation line in the object table.*
- class `__vector2list_stack_info`  
*A vector-to-list instrumentation line in the stack table.*
- struct `__warning_data`  
*Representation of a warning.*

#### Typedefs

- typedef std::vector  
    < \_\_cost\_factor \* > `__cost_factor_vector`
- typedef std::unordered\_map  
    < std::string, std::string > `__env_t`
- typedef void \* `__instruction_address_t`
- typedef const void \* `__object_t`
- typedef std::vector  
    < \_\_instruction\_address\_t > `__stack_npt`
- typedef `__stack_npt` \* `__stack_t`
- typedef std::vector  
    < `__warning_data` > `__warning_vector_t`

#### Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

## Functions

- `std::size_t __env_to_size_t (const char *__env_var, std::size_t __default_value)`
- `template<typename _InputIterator, typename _Function >`  
`_Function __for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `__stack_t __get_stack ()`
- `template<typename _Container >`  
`void __insert_top_n (_Container &__output, const typename _Container::value_type &__value, typename _Container::size_type __n)`
- `bool __is_invalid ()`
- `bool __is_off ()`
- `bool __is_on ()`
- `int __log2 (std::size_t __size)`
- `int __log_magnitude (float __f)`
- `float __map_erase_cost (std::size_t __size)`
- `float __map_find_cost (std::size_t __size)`
- `float __map_insert_cost (std::size_t __size)`
- `std::size_t __max_mem ()`
- `FILE * __open_output_file (const char *__extension)`
- `bool __profcxx_init ()`
- `void __profcxx_init_unconditional ()`
- `void __read_cost_factors ()`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator __remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `void __report (void)`
- `void __set_cost_factors ()`
- `void __set_max_mem ()`
- `void __set_max_stack_trace_depth ()`
- `void __set_max_warn_count ()`
- `void __set_trace_path ()`
- `std::size_t __size (__stack_t __stack)`
- `std::size_t __stack_max_depth ()`
- `template<typename _Container >`  
`void __top_n (const _Container &__input, _Container &__output, typename _Container::size_type __n)`
- `void __trace_hash_func_construct (const void *)`
- `void __trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __trace_hash_func_init ()`
- `void __trace_hash_func_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __trace_hashtable_size_construct (const void *, std::size_t)`
- `void __trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __trace_hashtable_size_init ()`
- `void __trace_hashtable_size_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_set_construct (const void *)`
- `void __trace_list_to_set_destruct (const void *)`
- `void __trace_list_to_set_find (const void *, std::size_t)`
- `void __trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_set_invalid_operator (const void *)`
- `void __trace_list_to_set_iterate (const void *, std::size_t)`
- `void __trace_list_to_slist_construct (const void *)`
- `void __trace_list_to_slist_destruct (const void *)`

- `void __trace_list_to_slist_init ()`
- `void __trace_list_to_slist_operation (const void *)`
- `void __trace_list_to_slist_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_list_to_slist_rewind (const void *)`
- `void __trace_list_to_vector_construct (const void *)`
- `void __trace_list_to_vector_destruct (const void *)`
- `void __trace_list_to_vector_init ()`
- `void __trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_vector_invalid_operator (const void *)`
- `void __trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __trace_list_to_vector_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_list_to_vector_resize (const void *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_construct (const void *)`
- `void __trace_map_to_unordered_map_destruct (const void *)`
- `void __trace_map_to_unordered_map_erase (const void *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_find (const void *, std::size_t)`
- `void __trace_map_to_unordered_map_init ()`
- `void __trace_map_to_unordered_map_insert (const void *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_invalidate (const void *)`
- `void __trace_map_to_unordered_map_iterate (const void *, std::size_t)`
- `void __trace_map_to_unordered_map_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_vector_size_construct (const void *, std::size_t)`
- `void __trace_vector_size_destruct (const void *, std::size_t, std::size_t)`
- `void __trace_vector_size_init ()`
- `void __trace_vector_size_report (FILE *, __warning_vector_t &)`
- `void __trace_vector_size_resize (const void *, std::size_t, std::size_t)`
- `void __trace_vector_to_list_construct (const void *)`
- `void __trace_vector_to_list_destruct (const void *)`
- `void __trace_vector_to_list_find (const void *, std::size_t)`
- `void __trace_vector_to_list_init ()`
- `void __trace_vector_to_list_insert (const void *, std::size_t, std::size_t)`
- `void __trace_vector_to_list_invalid_operator (const void *)`
- `void __trace_vector_to_list_iterate (const void *, std::size_t)`
- `void __trace_vector_to_list_report (FILE *, __warning_vector_t &)`
- `void __trace_vector_to_list_resize (const void *, std::size_t, std::size_t)`
- `bool __turn (__state_type __s)`
- `bool __turn_off ()`
- `bool __turn_on ()`
- `void __write (FILE * __f, __stack_t __stack)`
- `void __write_cost_factors ()`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __state, __INVALID)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_hash_func *, _S_hash_func, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_hashtable_size *, _S_hashtable_size, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_map2umap *, _S_map2umap, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_vector_size *, _S_vector_size, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_vector_to_list *, _S_vector_to_list, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_list_to_slist *, _S_list_to_slist, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_list_to_vector *, _S_list_to_vector, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __vector_shift_cost_factor, {"__vector_shift_cost_factor", 1.0})`

- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_iterate_cost_factor`,{"\_\_vector\_iterate\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_resize_cost_factor`,{"\_\_vector\_resize\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_shift_cost_factor`,{"\_\_list\_shift\_cost\_factor", 0.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_iterate_cost_factor`,{"\_\_list\_iterate\_cost\_factor", 10.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_resize_cost_factor`,{"\_\_list\_resize\_cost\_factor", 0.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_insert_cost_factor`,{"\_\_map\_insert\_cost\_factor", 1.5})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_erase_cost_factor`,{"\_\_map\_erase\_cost\_factor", 1.5})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_find_cost_factor`,{"\_\_map\_find\_cost\_factor", 1})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_iterate_cost_factor`,{"\_\_map\_iterate\_cost\_factor", 2.3})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_insert_cost_factor`,{"\_\_umap\_insert\_cost\_factor", 12.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_erase_cost_factor`,{"\_\_umap\_erase\_cost\_factor", 12.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_find_cost_factor`,{"\_\_umap\_find\_cost\_factor", 10.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_iterate_cost_factor`,{"\_\_umap\_iterate\_cost\_factor", 1.7})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor_vector` \*, `__cost_factors`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`const char` \*, `_S_trace_file_name`, `_GLIBCXX_PROFILE_TRACE_PATH_ROOT`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_warn_count`, `_GLIBCXX_PROFILE_MAX_WARN_COUNT`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_stack_depth`, `_GLIBCXX_PROFILE_MAX_STACK_DEPTH`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_mem`, `_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__env_t`, `__env`)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__gnu_cxx::__mutex`, `__global_lock`)

### 3.8.1 Detailed Description

GNU profile code for public use.

### 3.8.2 Typedef Documentation

#### 3.8.2.1 `typedef std::unordered_map<std::string, std::string> __gnu_profile::__env_t`

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is `libstdc++-profile.conf`. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 65 of file `profiler_trace.h`.

### 3.8.3 Function Documentation

#### 3.8.3.1 `bool __gnu_profile::__profcxx_init( ) [inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 649 of file `profiler_trace.h`.

#### 3.8.3.2 `void __gnu_profile::__report( void ) [inline]`

Final report method, registered with **atexit**.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in `profiler.h`. However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 440 of file `profiler_trace.h`.

References `std::min()`.

#### 3.8.3.3 `__gnu_profile::GLIBCXX_PROFILE_DEFINE_UNINIT_DATA( __gnu_cxx::__mutex , __global_lock )`

Master lock.

## 3.9 \_\_gnu\_sequential Namespace Reference

### 3.9.1 Detailed Description

GNU sequential classes for public use.

## 3.10 abi Namespace Reference

### 3.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`. A brief overview of an ABI is given in the `libstdc++` FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 3.11 std Namespace Reference

### Namespaces

- namespace [\\_\\_debug](#)
- namespace [\\_\\_detail](#)
- namespace [\\_\\_parallel](#)

- namespace [\\_\\_profile](#)
- namespace [chrono](#)
- namespace [decimal](#)
- namespace [placeholders](#)
- namespace [regex\\_constants](#)
- namespace [rel\\_ops](#)
- namespace [this\\_thread](#)
- namespace [tr1](#)
- namespace [tr2](#)

## Classes

- struct [\\_\\_atomic\\_base](#)  
*Base class for atomic integrals.*
- struct [\\_\\_atomic\\_base< \\_PTp \\* >](#)  
*Partial specialization for pointer types.*
- struct [\\_\\_atomic\\_flag\\_base](#)  
*Base type for atomic\_flag.*
- class [\\_\\_codecvt\\_abstract\\_base](#)  
*Common base for codecvt functions.*
- class [\\_\\_ctype\\_abstract\\_base](#)  
*Common base for ctype facet.*
- class [\\_\\_has\\_iterator\\_category\\_helper](#)  
*Traits class for iterators.*
- struct [\\_\\_is\\_location\\_invariant](#)
- struct [\\_\\_numeric\\_limits\\_base](#)  
*Part of std::numeric\_limits.*
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)  
*A deque::iterator.*
- struct [\\_Derives\\_from\\_binary\\_function](#)  
*Determines if the type \_Tp derives from binary\_function.*
- struct [\\_Derives\\_from\\_unary\\_function](#)  
*Determines if the type \_Tp derives from unary\_function.*
- class [\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [\\_Fwd\\_list\\_base](#)  
*Base class for forward\_list.*
- struct [\\_Fwd\\_list\\_const\\_iterator](#)  
*A forward\_list::const\_iterator.*
- struct [\\_Fwd\\_list\\_iterator](#)  
*A forward\_list::iterator.*
- struct [\\_Fwd\\_list\\_node](#)  
*A helper node class for forward\_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.*

- struct [\\_Fwd\\_list\\_node\\_base](#)  
*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*
- class [\\_Hashtable](#)
- class [\\_List\\_base](#)  
*See bits/stl\_deque.h's \_Deque\_base for an explanation.*
- struct [\\_List\\_const\\_iterator](#)  
*A list::const\_iterator.*
- struct [\\_List\\_iterator](#)  
*A list::iterator.*
- struct [\\_List\\_node](#)  
*An actual node in the list.*
- struct [\\_Maybe\\_get\\_result\\_type](#)  
*If we have found a result\_type, extract it.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_T1 >](#)  
*Derives from unary\_function, as appropriate.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_T1, \\_T2 >](#)  
*Derives from binary\_function, as appropriate.*
- struct [\\_Maybe\\_wrap\\_member\\_pointer](#)
- struct [\\_Maybe\\_wrap\\_member\\_pointer<\\_Tp \\_Class::\\* >](#)
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Implementation of mem\_fn for const member function pointers.*
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Implementation of mem\_fn for const volatile member function pointers.*
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Implementation of mem\_fn for volatile member function pointers.*
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) >](#)  
*Implementation of mem\_fn for member function pointers.*
- class [\\_Mu<\\_Arg, false, false >](#)
- class [\\_Mu<\\_Arg, false, true >](#)
- class [\\_Mu<\\_Arg, true, false >](#)
- class [\\_Mu<reference\\_wrapper<\\_Tp >, false, false >](#)
- struct [\\_Placeholder](#)  
*The type of placeholder objects defined by libstdc++.*
- struct [\\_Reference\\_wrapper\\_base](#)
- struct [\\_Safe\\_tuple\\_element](#)
- struct [\\_Safe\\_tuple\\_element\\_impl](#)
- struct [\\_Safe\\_tuple\\_element\\_impl<\\_\\_i, \\_Tuple, false >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Tuple\\_impl<\\_Idx >](#)
- struct [\\_Tuple\\_impl<\\_Idx, \\_Head, \\_Tail...>](#)
- struct [\\_Vector\\_base](#)  
*See bits/stl\_deque.h's \_Deque\_base for an explanation.*
- struct [\\_Weak\\_result\\_type](#)
- struct [\\_Weak\\_result\\_type\\_impl](#)
- struct [\\_Weak\\_result\\_type\\_impl<\\_Res\(&\)\(\\_ArgTypes...\) >](#)  
*Retrieve the result type for a function reference.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function type.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Retrieve result type for a const member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Retrieve result type for a const volatile member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Retrieve result type for a volatile member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve result type for a member function pointer.*
- struct [add\\_const](#)  
*add\_const*
- struct [add\\_cv](#)  
*add\_cv*
- struct [add\\_lvalue\\_reference](#)  
*add\_lvalue\_reference*
- struct [add\\_pointer](#)  
*add\_pointer*
- struct [add\\_rvalue\\_reference](#)  
*add\_rvalue\_reference*
- struct [add\\_volatile](#)  
*add\_volatile*
- struct [adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership and manage it.*
- struct [aligned\\_storage](#)  
*Alignment type.*
- struct [alignment\\_of](#)  
*alignment\_of*
- struct [allocator](#)  
*The standard allocator, as per [20.4].*
- class [allocator< void >](#)  
*allocator<void> specialization.*
- struct [allocator\\_arg\\_t](#)  
*[allocator.tag]*
- struct [allocator\\_traits](#)  
*Uniform interface to all allocator types.*
- struct [array](#)  
*A standard container for storing a fixed size sequence of elements.*
- struct [atomic](#)  
*Generic atomic type, primary class template.*
- struct [atomic< \\_Tp \\* >](#)  
*Partial specialization for pointer types.*
- struct [atomic< bool >](#)  
*Explicit specialization for bool.*
- struct [atomic< char >](#)

- Explicit specialization for char.*
- struct `atomic< char16_t >`
  - Explicit specialization for char16\_t.*
- struct `atomic< char32_t >`
  - Explicit specialization for char32\_t.*
- struct `atomic< int >`
  - Explicit specialization for int.*
- struct `atomic< long >`
  - Explicit specialization for long.*
- struct `atomic< long long >`
  - Explicit specialization for long long.*
- struct `atomic< short >`
  - Explicit specialization for short.*
- struct `atomic< signed char >`
  - Explicit specialization for signed char.*
- struct `atomic< unsigned char >`
  - Explicit specialization for unsigned char.*
- struct `atomic< unsigned int >`
  - Explicit specialization for unsigned int.*
- struct `atomic< unsigned long >`
  - Explicit specialization for unsigned long.*
- struct `atomic< unsigned long long >`
  - Explicit specialization for unsigned long long.*
- struct `atomic< unsigned short >`
  - Explicit specialization for unsigned short.*
- struct `atomic< wchar_t >`
  - Explicit specialization for wchar\_t.*
- struct `atomic_bool`
  - atomic\_bool*
- struct `atomic_flag`
  - atomic\_flag*
- class `auto_ptr`
  - A simple smart pointer providing strict ownership semantics.*
- struct `auto_ptr_ref`
- class `back_insert_iterator`
  - Turns assignment into insertion.*
- class `bad_alloc`
  - Exception possibly thrown by new.*
  - bad\_alloc (or classes derived from it) is used to report allocation errors from the throwing forms of new.*
- class `bad_cast`
  - Thrown during incorrect typecasting.*
  - If you attempt an invalid dynamic\_cast expression, an instance of this class (or something derived from this class) is thrown.*
- class `bad_exception`
- class `bad_function_call`
  - Exception class thrown when class template function's operator() is called with an empty target.*
- class `bad_typeid`
  - Thrown when a NULL pointer in a typeid expression is used.*

- class [bad\\_weak\\_ptr](#)  
*Exception possibly thrown by `shared_ptr`.*
- class [basic\\_filebuf](#)  
*The actual work of input and output (for files).*
- class [basic\\_fstream](#)  
*Controlling input and output for files.*
- class [basic\\_ifstream](#)  
*Controlling input for files.*
- class [basic\\_ios](#)  
*Template class `basic_ios`, virtual base class for all stream classes.*
- class [basic\\_iostream](#)  
*Template class `basic_iostream`.*
- class [basic\\_istream](#)  
*Template class `basic_istream`.*
- class [basic\\_istreambuf](#)  
*Controlling input for `std::string`.*
- class [basic\\_ofstream](#)  
*Controlling output for files.*
- class [basic\\_ostream](#)  
*Template class `basic_ostream`.*
- class [basic\\_ostringstream](#)  
*Controlling output for `std::string`.*
- class [basic\\_regex](#)
- class [basic\\_streambuf](#)  
*The actual work of input and output (interface).*
- class [basic\\_string](#)  
*Managing sequences of characters and character-like objects.*
- class [basic\\_stringbuf](#)  
*The actual work of input and output (for `std::string`).*
- class [basic\\_stringstream](#)  
*Controlling input and output for `std::string`.*
- class [bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- struct [bidirectional\\_iterator\\_tag](#)  
*Bidirectional iterators support a superset of forward iterator operations.*
- struct [binary\\_function](#)
- class [binary\\_negate](#)  
*One of the [negation functors](#).*
- class [binder1st](#)  
*One of the [binder functors](#).*
- class [binder2nd](#)  
*One of the [binder functors](#).*
- class [binomial\\_distribution](#)  
*A discrete binomial random number distribution.*
- class [cauchy\\_distribution](#)  
*A `cauchy_distribution` random number distribution.*
- struct [char\\_traits](#)

- Basis for explicit traits specializations.*
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)
  - char\_traits<\_\_gnu\_cxx::character> specialization.*
- struct [char\\_traits< char >](#)
  - 21.1.3.1 char\_traits specializations*
- struct [char\\_traits< wchar\\_t >](#)
  - 21.1.3.2 char\_traits specializations*
- class [chi\\_squared\\_distribution](#)
  - A chi\_squared\_distribution random number distribution.*
- class [codecvt](#)
  - Primary class template codecvt.*
  - NB: Generic, mostly useless implementation.*
- class [codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#)
  - codecvt<InternT, \_ExternT, encoding\_state> specialization.*
- class [codecvt< char, char, mbstate\\_t >](#)
  - class codecvt<char, char, mbstate\_t> specialization.*
- class [codecvt< wchar\\_t, char, mbstate\\_t >](#)
  - class codecvt<wchar\_t, char, mbstate\_t> specialization.*
- class [codecvt\\_base](#)
  - Empty base class for codecvt facet [22.2.1.5].*
- class [codecvt\\_byname](#)
  - class codecvt\_byname [22.2.1.6].*
- class [collate](#)
  - Facet for localized string comparison.*
- class [collate\\_byname](#)
  - class collate\_byname [22.2.4.2].*
- struct [complex](#)
- struct [complex< double >](#)
  - 26.2.3 complex specializations complex<double> specialization*
- struct [complex< float >](#)
  - 26.2.3 complex specializations complex<float> specialization*
- struct [complex< long double >](#)
  - 26.2.3 complex specializations complex<long double> specialization*
- class [condition\\_variable](#)
  - condition\_variable*
- class [condition\\_variable\\_any](#)
  - condition\_variable\_any*
- struct [conditional](#)
  - Define a member typedef type to one of two argument types.*
- class [const\\_mem\\_fun1\\_ref\\_t](#)
  - One of the [adaptors for member pointers](#).*
- class [const\\_mem\\_fun1\\_t](#)
  - One of the [adaptors for member pointers](#).*
- class [const\\_mem\\_fun\\_ref\\_t](#)
  - One of the [adaptors for member pointers](#).*
- class [const\\_mem\\_fun\\_t](#)
  - One of the [adaptors for member pointers](#).*

- class [ctype](#)

*Primary class template ctype facet.*  
*This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class [ctype< char >](#)

*The ctype<char> specialization.*  
*This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.*
- class [ctype< wchar\\_t >](#)

*The ctype<wchar\_t> specialization.*  
*This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.*
- struct [ctype\\_base](#)

*Base class for ctype.*
- class [ctype\\_byname](#)

*class ctype\_byname [22.2.1.2].*
- class [ctype\\_byname< char >](#)

*22.2.1.4 Class ctype\_byname specializations.*
- class [decay](#)

*decay*
- struct [default\\_delete](#)

*Primary template, default\_delete.*
- struct [default\\_delete< \\_Tp\[\]>](#)

*Specialization, default\_delete.*
- struct [defer\\_lock\\_t](#)

*Do not acquire ownership of the mutex.*
- class [deque](#)

*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*
- class [discard\\_block\\_engine](#)
- class [discrete\\_distribution](#)

*A discrete\_distribution random number distribution.*
- struct [divides](#)

*One of the [math functors](#).*
- class [domain\\_error](#)
- struct [enable\\_if](#)

*Define a member typedef `type` only if a boolean constant is true.*
- class [enable\\_shared\\_from\\_this](#)

*Base class allowing use of member function `shared_from_this`.*
- struct [equal\\_to](#)

*One of the [comparison functors](#).*
- class [error\\_category](#)

*error\_category*
- struct [error\\_code](#)

*error\_code*
- struct [error\\_condition](#)

*error\_condition*
- class [exception](#)

*Base class for all library exceptions.*

- class [exponential\\_distribution](#)  
*An exponential continuous distribution for random numbers.*
- struct [extent](#)  
*extent*
- class [extreme\\_value\\_distribution](#)  
*A extreme\_value\_distribution random number distribution.*
- class [fisher\\_f\\_distribution](#)  
*A fisher\_f\_distribution random number distribution.*
- struct [forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- class [forward\\_list](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*
- class [fpos](#)  
*Class representing stream positions.*
- class [front\\_insert\\_iterator](#)  
*Turns assignment into insertion.*
- class [function< \\_Res\( \\_ArgTypes...\)>](#)  
*Primary class template for std::function.  
Polymorphic function wrapper.*
- class [future\\_error](#)  
*Exception type thrown by futures.*
- class [gamma\\_distribution](#)  
*A gamma continuous distribution for random numbers.*
- class [geometric\\_distribution](#)  
*A discrete geometric random number distribution.*
- struct [greater](#)  
*One of the [comparison functors](#).*
- struct [greater\\_equal](#)  
*One of the [comparison functors](#).*
- class [gslice](#)  
*Class defining multi-dimensional subset of an array.*
- class [gslice\\_array](#)  
*Reference to multi-dimensional subset of an array.*
- struct [has\\_trivial\\_copy\\_assign](#)  
*has\_trivial\_copy\_assign (temporary legacy)*
- struct [has\\_trivial\\_copy\\_constructor](#)  
*has\_trivial\_copy\_constructor (temporary legacy)*
- struct [has\\_trivial\\_default\\_constructor](#)  
*has\_trivial\_default\_constructor (temporary legacy)*
- struct [has\\_virtual\\_destructor](#)  
*has\_virtual\_destructor*
- struct [hash< \\_\\_debug::bitset< \\_Nb > >](#)  
*std::hash specialization for [bitset](#).*
- struct [hash< \\_\\_debug::vector< bool, \\_Alloc > >](#)  
*std::hash specialization for [vector< bool >](#).*
- struct [hash< \\_\\_gnu\\_cxx::\\_\\_u16vstring >](#)  
*std::hash specialization for [\\_\\_u16vstring](#).*

- struct `hash< __gnu_cxx::__u32vstring >`  
*std::hash specialization for \_\_u32vstring.*
- struct `hash< __gnu_cxx::__vstring >`  
*std::hash specialization for \_\_vstring.*
- struct `hash< __gnu_cxx::__wvstring >`  
*std::hash specialization for \_\_wvstring.*
- struct `hash< __gnu_cxx::throw_value_limit >`  
*Explicit specialization of std::hash for \_\_gnu\_cxx::throw\_value\_limit.*
- struct `hash< __gnu_cxx::throw_value_random >`  
*Explicit specialization of std::hash for \_\_gnu\_cxx::throw\_value\_limit.*
- struct `hash< __profile::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash< __profile::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- struct `hash< __shared_ptr< _Tp, _Lp > >`  
*std::hash specialization for \_\_shared\_ptr.*
- struct `hash< _Tp * >`  
*Partial specializations for pointer types.*
- struct `hash< bool >`  
*Explicit specialization for bool.*
- struct `hash< char >`  
*Explicit specialization for char.*
- struct `hash< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `hash< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `hash< double >`  
*Specialization for double.*
- struct `hash< error_code >`  
*std::hash specialization for error\_code.*
- struct `hash< float >`  
*Specialization for float.*
- struct `hash< int >`  
*Explicit specialization for int.*
- struct `hash< long >`  
*Explicit specialization for long.*
- struct `hash< long double >`  
*Specialization for long double.*
- struct `hash< long long >`  
*Explicit specialization for long long.*
- struct `hash< shared_ptr< _Tp > >`  
*std::hash specialization for shared\_ptr.*
- struct `hash< short >`  
*Explicit specialization for short.*
- struct `hash< signed char >`  
*Explicit specialization for signed char.*
- struct `hash< string >`

- std::hash specialization for string.*
- struct [hash< thread::id >](#)
  - std::hash specialization for thread::id.*
- struct [hash< type\\_index >](#)
  - std::hash specialization for type\_index.*
- struct [hash< u16string >](#)
  - std::hash specialization for u16string.*
- struct [hash< u32string >](#)
  - std::hash specialization for u32string.*
- struct [hash< unique\\_ptr< \\_Tp, \\_Dp > >](#)
  - std::hash specialization for unique\_ptr.*
- struct [hash< unsigned char >](#)
  - Explicit specialization for unsigned char.*
- struct [hash< unsigned int >](#)
  - Explicit specialization for unsigned int.*
- struct [hash< unsigned long >](#)
  - Explicit specialization for unsigned long.*
- struct [hash< unsigned long long >](#)
  - Explicit specialization for unsigned long long.*
- struct [hash< unsigned short >](#)
  - Explicit specialization for unsigned short.*
- struct [hash< wchar\\_t >](#)
  - Explicit specialization for wchar\_t.*
- struct [hash< wstring >](#)
  - std::hash specialization for wstring.*
- struct [hash<::bitset< \\_Nb > >](#)
  - std::hash specialization for bitset.*
- struct [hash<::vector< bool, \\_Alloc > >](#)
  - std::hash specialization for vector<bool>.*
- class [independent\\_bits\\_engine](#)
- class [indirect\\_array](#)
  - Reference to arbitrary subset of an array.*
- class [initializer\\_list](#)
  - initializer\_list*
- struct [input\\_iterator\\_tag](#)
  - Marking input iterators.*
- class [insert\\_iterator](#)
  - Turns assignment into insertion.*
- struct [integral\\_constant](#)
  - integral\_constant*
- class [invalid\\_argument](#)
- class [ios\\_base](#)
  - The base of the I/O class hierarchy.*
  - This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see ios\_base when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- struct [is\\_abstract](#)
  - is\_abstract*

- struct [is\\_arithmetic](#)  
*is\_arithmetic*
- struct [is\\_array](#)  
*is\_array*
- struct [is\\_assignable](#)  
*is\_assignable*
- struct [is\\_base\\_of](#)  
*is\_base\_of*
- struct [is\\_bind\\_expression](#)  
*Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].*
- struct [is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind_result` is always a bind expression.*
- struct [is\\_bind\\_expression< const \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression< const \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind_result` is always a bind expression.*
- struct [is\\_bind\\_expression< const volatile \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression< const volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind_result` is always a bind expression.*
- struct [is\\_bind\\_expression< volatile \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression< volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind_result` is always a bind expression.*
- struct [is\\_class](#)  
*is\_class*
- struct [is\\_compound](#)  
*is\_compound*
- struct [is\\_const](#)  
*is\_const*
- struct [is\\_constructible](#)  
*is\_constructible*
- struct [is\\_convertible](#)  
*is\_convertible*
- struct [is\\_copy\\_assignable](#)  
*is\_copy\_assignable*
- struct [is\\_copy\\_constructible](#)  
*is\_copy\_constructible*
- struct [is\\_default\\_constructible](#)  
*is\_default\_constructible*
- struct [is\\_destructible](#)  
*is\_destructible*
- struct [is\\_empty](#)  
*is\_empty*

- struct [is\\_enum](#)  
*is\_enum*
- struct [is\\_error\\_code\\_enum](#)  
*is\_error\_code\_enum*
- struct [is\\_error\\_code\\_enum< future\\_errc >](#)  
*Specialization.*
- struct [is\\_error\\_condition\\_enum](#)  
*is\_error\_condition\_enum*
- struct [is\\_floating\\_point](#)  
*is\_floating\_point*
- struct [is\\_function](#)  
*is\_function*
- struct [is\\_fundamental](#)  
*is\_fundamental*
- struct [is\\_integral](#)  
*is\_integral*
- struct [is\\_literal\\_type](#)  
*is\_literal\_type*
- struct [is\\_lvalue\\_reference](#)  
*is\_lvalue\_reference*
- struct [is\\_member\\_function\\_pointer](#)  
*is\_member\_function\_pointer*
- struct [is\\_member\\_object\\_pointer](#)  
*is\_member\_object\_pointer*
- struct [is\\_member\\_pointer](#)  
*is\_member\_pointer*
- struct [is\\_move\\_assignable](#)  
*is\_move\_assignable*
- struct [is\\_move\\_constructible](#)  
*is\_move\_constructible*
- struct [is\\_nothrow\\_assignable](#)  
*is\_nothrow\_assignable*
- struct [is\\_nothrow\\_constructible](#)  
*is\_nothrow\_constructible*
- struct [is\\_nothrow\\_copy\\_assignable](#)  
*is\_nothrow\_copy\_assignable*
- struct [is\\_nothrow\\_copy\\_constructible](#)  
*is\_nothrow\_copy\_constructible*
- struct [is\\_nothrow\\_default\\_constructible](#)  
*is\_nothrow\_default\_constructible*
- struct [is\\_nothrow\\_destructible](#)  
*is\_nothrow\_destructible*
- struct [is\\_nothrow\\_move\\_assignable](#)  
*is\_nothrow\_move\_assignable*
- struct [is\\_nothrow\\_move\\_constructible](#)  
*is\_nothrow\_move\_constructible*
- struct [is\\_object](#)

- is\_object*
- struct [is\\_placeholder](#)
  - Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct [is\\_placeholder< \\_Placeholder< \\_Num > >](#)
- struct [is\\_pod](#)
  - is\_pod*
- struct [is\\_pointer](#)
  - is\_pointer*
- struct [is\\_polymorphic](#)
  - is\_polymorphic*
- struct [is\\_reference](#)
  - is\_reference*
- struct [is\\_rvalue\\_reference](#)
  - is\_rvalue\_reference*
- struct [is\\_same](#)
  - is\_same*
- struct [is\\_scalar](#)
  - is\_scalar*
- struct [is\\_signed](#)
  - is\_signed*
- struct [is\\_standard\\_layout](#)
  - is\_standard\_layout*
- struct [is\\_trivial](#)
  - is\_trivial*
- struct [is\\_trivially\\_destructible](#)
  - is\_trivially\_constructible (still unimplemented)*
- struct [is\\_union](#)
  - is\_union*
- struct [is\\_unsigned](#)
  - is\_unsigned*
- struct [is\\_void](#)
  - is\_void*
- struct [is\\_volatile](#)
  - is\_volatile*
- class [istream\\_iterator](#)
  - Provides input iterator semantics for streams.*
- class [istreambuf\\_iterator](#)
  - Provides input iterator semantics for streambufs.*
- struct [iterator](#)
  - Common iterator class.*
- struct [iterator\\_traits< \\_Tp \\* >](#)
  - Partial specialization for pointer types.*
- struct [iterator\\_traits< const \\_Tp \\* >](#)
  - Partial specialization for const pointer types.*
- class [length\\_error](#)
- struct [less](#)
  - One of the [comparison functors](#).*

- struct [less\\_equal](#)  
*One of the [comparison functors](#).*
- class [linear\\_congruential\\_engine](#)  
*A model of a linear congruential random number generator.*
- class [list](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*
- class [locale](#)  
*Container class for localization functionality.  
The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization.  
A locale is a collection of facets that implement various localization features such as money, time, and number printing.*
- class [lock\\_guard](#)  
*Scoped lock idiom.*
- class [logic\\_error](#)  
*One of two subclasses of exception.*
- struct [logical\\_and](#)  
*One of the [Boolean operations functors](#).*
- struct [logical\\_not](#)  
*One of the [Boolean operations functors](#).*
- struct [logical\\_or](#)  
*One of the [Boolean operations functors](#).*
- class [lognormal\\_distribution](#)  
*A lognormal\_distribution random number distribution.*
- struct [make\\_signed](#)  
*make\_signed*
- struct [make\\_unsigned](#)  
*make\_unsigned*
- class [map](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- class [mask\\_array](#)  
*Reference to selected subset of an array.*
- class [match\\_results](#)  
*The results of a match or search operation.*
- class [mem\\_fun1\\_ref\\_t](#)  
*One of the [adaptors for member pointers](#).*
- class [mem\\_fun1\\_t](#)  
*One of the [adaptors for member pointers](#).*
- class [mem\\_fun\\_ref\\_t](#)  
*One of the [adaptors for member pointers](#).*
- class [mem\\_fun\\_t](#)  
*One of the [adaptors for member pointers](#).*
- class [mersenne\\_twister\\_engine](#)
- class [messages](#)  
*Primary class template messages.  
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [messages\\_base](#)  
*Messages facet base class providing catalog typedef.*
- class [messages\\_byname](#)

- class messages\_byname [22.2.7.2].*
- struct [minus](#)
  - One of the [math functors](#).*
- struct [modulus](#)
  - One of the [math functors](#).*
- class [money\\_base](#)
  - Money format ordering data.*
  - This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*
- class [money\\_get](#)
  - Primary class template money\_get.*
  - This facet encapsulates the code to parse and return a monetary amount from a string.*
- class [money\\_put](#)
  - Primary class template money\_put.*
  - This facet encapsulates the code to format and output a monetary amount.*
- class [moneypunct](#)
  - Primary class template moneypunct.*
  - This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*
- class [moneypunct\\_byname](#)
  - class moneypunct\_byname [22.2.6.4].*
- class [move\\_iterator](#)
- class [multimap](#)
  - A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- struct [multiplies](#)
  - One of the [math functors](#).*
- class [multiset](#)
  - A standard container made up of elements, which can be retrieved in logarithmic time.*
- class [mutex](#)
  - mutex*
- struct [negate](#)
  - One of the [math functors](#).*
- class [negative\\_binomial\\_distribution](#)
  - A negative\_binomial\_distribution random number distribution.*
- class [nested\\_exception](#)
  - Exception class with exception\_ptr data member.*
- class [normal\\_distribution](#)
  - A normal continuous distribution for random numbers.*
- struct [not\\_equal\\_to](#)
  - One of the [comparison functors](#).*
- class [num\\_get](#)
  - Primary class template num\_get.*
  - This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class [num\\_put](#)
  - Primary class template num\_put.*
  - This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- struct [numeric\\_limits](#)
  - Properties of fundamental types.*

- struct [numeric\\_limits< bool >](#)  
*numeric\_limits<bool> specialization.*
- struct [numeric\\_limits< char >](#)  
*numeric\_limits<char> specialization.*
- struct [numeric\\_limits< char16\\_t >](#)  
*numeric\_limits<char16\_t> specialization.*
- struct [numeric\\_limits< char32\\_t >](#)  
*numeric\_limits<char32\_t> specialization.*
- struct [numeric\\_limits< double >](#)  
*numeric\_limits<double> specialization.*
- struct [numeric\\_limits< float >](#)  
*numeric\_limits<float> specialization.*
- struct [numeric\\_limits< int >](#)  
*numeric\_limits<int> specialization.*
- struct [numeric\\_limits< long >](#)  
*numeric\_limits<long> specialization.*
- struct [numeric\\_limits< long double >](#)  
*numeric\_limits<long double> specialization.*
- struct [numeric\\_limits< long long >](#)  
*numeric\_limits<long long> specialization.*
- struct [numeric\\_limits< short >](#)  
*numeric\_limits<short> specialization.*
- struct [numeric\\_limits< signed char >](#)  
*numeric\_limits<signed char> specialization.*
- struct [numeric\\_limits< unsigned char >](#)  
*numeric\_limits<unsigned char> specialization.*
- struct [numeric\\_limits< unsigned int >](#)  
*numeric\_limits<unsigned int> specialization.*
- struct [numeric\\_limits< unsigned long >](#)  
*numeric\_limits<unsigned long> specialization.*
- struct [numeric\\_limits< unsigned long long >](#)  
*numeric\_limits<unsigned long long> specialization.*
- struct [numeric\\_limits< unsigned short >](#)  
*numeric\_limits<unsigned short> specialization.*
- struct [numeric\\_limits< wchar\\_t >](#)  
*numeric\_limits<wchar\_t> specialization.*
- class [numpunct](#)  
*Primary class template numpunct.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class [numpunct\\_byname](#)  
*class numpunct\_byname [22.2.3.2].*
- struct [once\\_flag](#)  
*once\_flag*
- class [ostream\\_iterator](#)  
*Provides output iterator semantics for streams.*
- class [ostreambuf\\_iterator](#)

*Provides output iterator semantics for streambufs.*

- class [out\\_of\\_range](#)
- struct [output\\_iterator\\_tag](#)

*Marking output iterators.*

- class [overflow\\_error](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)

*Partial specialization of owner\_less for shared\_ptr.*

- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)

*Partial specialization of owner\_less for weak\_ptr.*

- struct [pair](#)

*Struct holding two objects of arbitrary type.*

- class [piecewise\\_constant\\_distribution](#)
- struct [piecewise\\_construct\\_t](#)

*piecewise\_construct\_t*

- class [piecewise\\_linear\\_distribution](#)
- struct [plus](#)

*A piecewise\_linear\_distribution random number distribution.*

*One of the math functors.*

- class [pointer\\_to\\_binary\\_function](#)
- class [pointer\\_to\\_unary\\_function](#)

*One of the adaptors for function pointers.*

- struct [pointer\\_traits](#)

*Uniform interface to all pointer-like types.*

- struct [pointer\\_traits< \\_Tp \\* >](#)

*Partial specialization for built-in pointers.*

- class [poisson\\_distribution](#)
- class [priority\\_queue](#)

*A discrete Poisson random number distribution.*

- class [queue](#)
- struct [random\\_access\\_iterator\\_tag](#)

*A standard container giving FIFO behavior.*

*Random-access iterators support a superset of bidirectional iterator operations.*

- class [random\\_device](#)
- class [range\\_error](#)
- struct [rank](#)

*rank*

- struct [ratio](#)

*Provides compile-time rational arithmetic.*

- struct [ratio\\_equal](#)

*ratio\_equal*

- struct [ratio\\_not\\_equal](#)

*ratio\_not\_equal*

- class [raw\\_storage\\_iterator](#)
- class [recursive\\_mutex](#)

- recursive\_mutex*
- class [reference\\_wrapper](#)
  - Primary class template for reference\_wrapper.*
- class [regex\\_error](#)
  - A regular expression exception class.*
  - The regular expression library throws objects of this class on error.*
- class [regex\\_iterator](#)
- class [regex\\_token\\_iterator](#)
- struct [regex\\_traits](#)
  - Class regex\_traits. Describes aspects of a regular expression.*
- struct [remove\\_all\\_extents](#)
  - remove\_all\_extents*
- struct [remove\\_const](#)
  - remove\_const*
- struct [remove\\_cv](#)
  - remove\_cv*
- struct [remove\\_extent](#)
  - remove\_extent*
- struct [remove\\_pointer](#)
  - remove\_pointer*
- struct [remove\\_reference](#)
  - remove\_reference*
- struct [remove\\_volatile](#)
  - remove\_volatile*
- class [reverse\\_iterator](#)
- class [runtime\\_error](#)
  - One of two subclasses of exception.*
- class [scoped\\_allocator\\_adaptor](#)
  - Primary class template.*
- class [seed\\_seq](#)
  - The seed\_seq class generates sequences of seeds for random number generators.*
- class [set](#)
  - A standard container made up of unique keys, which can be retrieved in logarithmic time.*
- class [shared\\_ptr](#)
  - A smart pointer with reference-counted copy semantics.*
- class [shuffle\\_order\\_engine](#)
  - Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `___w`.*
- class [slice](#)
  - Class defining one-dimensional subset of an array.*
- class [slice\\_array](#)
  - Reference to one-dimensional subset of an array.*
- class [stack](#)
  - A standard container giving FILO behavior.*
- class [student\\_t\\_distribution](#)
  - A student\_t\_distribution random number distribution.*
- class [sub\\_match](#)
- class [system\\_error](#)

- Thrown to indicate error code of underlying system.*

  - class [thread](#)
    - thread*
  - class [time\\_base](#)
    - Time format ordering data.*
    - This class provides an enum representing different orderings of time: day, month, and year.*
  - class [time\\_get](#)
    - Primary class template time\_get.*
    - This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*
  - class [time\\_get\\_byname](#)
    - class time\_get\_byname [22.2.5.2].*
  - class [time\\_put](#)
    - Primary class template time\_put.*
    - This facet encapsulates the code to format and output dates and times according to formats used by strftime().*
  - class [time\\_put\\_byname](#)
    - class time\_put\_byname [22.2.5.4].*
  - struct [try\\_to\\_lock\\_t](#)
    - Try to acquire ownership of the mutex without blocking.*
  - class [tuple](#)
    - Primary class template, tuple.*
  - class [tuple<\\_T1, \\_T2>](#)
    - Partial specialization, 2-element tuple. Includes construction and assignment from a pair.*
  - struct [tuple\\_element<0, tuple<\\_Head, \\_Tail...>>](#)
  - struct [tuple\\_element<\\_i, tuple<\\_Head, \\_Tail...>>](#)
  - struct [tuple\\_size<tuple<\\_Elements...>>](#)
    - class tuple\_size*
  - struct [type\\_index](#)
    - Class type\_index*
    - The class type\_index provides a simple wrapper for type\_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).*
  - class [type\\_info](#)
    - Part of RTTI.*
  - struct [unary\\_function](#)
  - class [unary\\_negate](#)
    - One of the [negation functors](#).*
  - class [underflow\\_error](#)
  - struct [underlying\\_type](#)
    - The underlying type of an enum.*
  - class [uniform\\_int\\_distribution](#)
    - Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
  - class [uniform\\_real\\_distribution](#)
    - Uniform continuous distribution for random numbers.*
  - class [unique\\_lock](#)
    - unique\_lock*
  - class [unique\\_ptr](#)
    - 20.7.1.2 unique\_ptr for single objects.*
  - class [unique\\_ptr<\\_Tp\[\], \\_Dp>](#)

20.7.1.3 *unique\_ptr* for array objects with a runtime length

- class [unordered\\_map](#)  
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class [unordered\\_multimap](#)  
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class [unordered\\_multiset](#)  
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class [unordered\\_set](#)  
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.
- struct [uses\\_allocator](#)  
*[allocator.uses.trait]*
- struct [uses\\_allocator](#)< tuple< \_Types...>, \_Alloc >  
Partial specialization for tuples.
- class [valarray](#)  
Smart array designed to support numeric processing.
- class [vector](#)  
A standard container which offers fixed time access to individual elements in any order.
- class [vector](#)< bool, \_Alloc >  
A specialization of vector for booleans which offers fixed time access to individual elements in any order.
- class [weak\\_ptr](#)  
A smart pointer with weak semantics.
- class [weibull\\_distribution](#)  
A weibull\_distribution random number distribution.

## Typedefs

- template<typename \_Tp >  
using [\\_\\_allocator\\_base](#) = [\\_\\_gnu\\_cxx::new\\_allocator](#)< \_Tp >
- typedef unsigned char [\\_\\_atomic\\_flag\\_data\\_type](#)
- typedef FILE [\\_\\_c\\_file](#)
- typedef \_\_locale\_t [\\_\\_c\\_locale](#)
- typedef \_\_gthread\_mutex\_t [\\_\\_c\\_lock](#)
- template<typename \_Tp, typename \_Hash >  
using [\\_\\_cache\\_default](#) = \_\_not\_< \_\_and\_< \_\_is\_fast\_hash< \_Hash >, [is\\_default\\_constructible](#)< \_Hash >, [is\\_copy\\_assignable](#)< \_Hash >, \_\_detail::\_\_is\_noexcept\_hash< \_Tp, \_Hash >>>
- template<typename \_Alloc >  
using [\\_\\_check\\_copy\\_constructible](#) = \_\_allow\_copy\_cons< \_\_is\_copy\_insertable< \_Alloc >::value >
- template<typename \_Tp >  
using [\\_\\_empty\\_not\\_final](#) = typename [conditional](#)< \_\_is\_final(\_Tp), [false\\_type](#), \_\_is\_empty\_non\_tuple< \_Tp >>::type
- template<typename \_Tp, typename \_Tp2 = typename decay<\_Tp>::type>  
using [\\_\\_is\\_socketlike](#) = \_\_or\_< [is\\_integral](#)< \_Tp2 >, [is\\_enum](#)< \_Tp2 >>
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc >  
using [\\_\\_sub\\_match\\_string](#) = [basic\\_string](#)< typename iterator\_traits< \_Bi\_iter >::value\_type, \_Ch\_traits, \_Ch\_alloc >

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>  
using __umap_hashtable = __Hashtable<_Key, std::pair<const _Key, _Tp>, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr>`
- `template<bool _Cache>  
using __umap_traits = __detail::__Hashtable_traits<_Cache, false, true>`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>  
using __ummap_hashtable = __Hashtable<_Key, std::pair<const _Key, _Tp>, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr>`
- `template<bool _Cache>  
using __ummap_traits = __detail::__Hashtable_traits<_Cache, false, false>`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>  
using __umset_hashtable = __Hashtable<_Value, _Value, _Alloc, __detail::__Identity, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr>`
- `template<bool _Cache>  
using __umset_traits = __detail::__Hashtable_traits<_Cache, true, false>`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>  
using __uset_hashtable = __Hashtable<_Value, _Value, _Alloc, __detail::__Identity, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr>`
- `template<bool _Cache>  
using __uset_traits = __detail::__Hashtable_traits<_Cache, true, true>`
- `typedef unsigned long __Bit_type`
- `template<typename _Tp1, typename _Tp2>  
using __NotSame = __not< is_same<typename std::decay<_Tp1>::type, typename std::decay<_Tp2>::type>>`
- `template<typename... _Cond>  
using __Require = typename enable_if<__and<_Cond...>::value>::type`
- `template<typename _InIter>  
using __RequireInputIter = typename enable_if<is_convertible<typename iterator_traits<_InIter>::iterator_category, input_iterator_tag>::value>::type`
- `typedef __atomic_base<char> atomic_char`
- `typedef __atomic_base<char16_t> atomic_char16_t`
- `typedef __atomic_base<char32_t> atomic_char32_t`
- `typedef __atomic_base<int> atomic_int`
- `typedef __atomic_base  
<int_fast16_t> atomic_int_fast16_t`
- `typedef __atomic_base  
<int_fast32_t> atomic_int_fast32_t`
- `typedef __atomic_base  
<int_fast64_t> atomic_int_fast64_t`
- `typedef __atomic_base  
<int_fast8_t> atomic_int_fast8_t`
- `typedef __atomic_base  
<int_least16_t> atomic_int_least16_t`
- `typedef __atomic_base  
<int_least32_t> atomic_int_least32_t`
- `typedef __atomic_base  
<int_least64_t> atomic_int_least64_t`

- typedef [\\_\\_atomic\\_base](#)  
    < int\_least8\_t > [atomic\\_int\\_least8\\_t](#)
- typedef [\\_\\_atomic\\_base](#)< intmax\_t > [atomic\\_intmax\\_t](#)
- typedef [\\_\\_atomic\\_base](#)< intptr\_t > [atomic\\_intptr\\_t](#)
- typedef [\\_\\_atomic\\_base](#)< long long > [atomic\\_llong](#)
- typedef [\\_\\_atomic\\_base](#)< long > [atomic\\_long](#)
- typedef [\\_\\_atomic\\_base](#)< ptrdiff\_t > [atomic\\_ptrdiff\\_t](#)
- typedef [\\_\\_atomic\\_base](#)< signed  
    char > [atomic\\_schar](#)
- typedef [\\_\\_atomic\\_base](#)< short > [atomic\\_short](#)
- typedef [\\_\\_atomic\\_base](#)< size\_t > [atomic\\_size\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < unsigned char > [atomic\\_uchar](#)
- typedef [\\_\\_atomic\\_base](#)  
    < unsigned int > [atomic\\_uint](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_fast16\_t > [atomic\\_uint\\_fast16\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_fast32\_t > [atomic\\_uint\\_fast32\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_fast64\_t > [atomic\\_uint\\_fast64\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_fast8\_t > [atomic\\_uint\\_fast8\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_least16\_t > [atomic\\_uint\\_least16\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_least32\_t > [atomic\\_uint\\_least32\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_least64\_t > [atomic\\_uint\\_least64\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < uint\_least8\_t > [atomic\\_uint\\_least8\\_t](#)
- typedef [\\_\\_atomic\\_base](#)< uintmax\_t > [atomic\\_uintmax\\_t](#)
- typedef [\\_\\_atomic\\_base](#)< uintptr\_t > [atomic\\_uintptr\\_t](#)
- typedef [\\_\\_atomic\\_base](#)  
    < unsigned long long > [atomic\\_ullong](#)
- typedef [\\_\\_atomic\\_base](#)  
    < unsigned long > [atomic\\_ulong](#)
- typedef [\\_\\_atomic\\_base](#)  
    < unsigned short > [atomic\\_ushort](#)
- typedef [\\_\\_atomic\\_base](#)< wchar\_t > [atomic\\_wchar\\_t](#)
- typedef [match\\_results](#)< const  
    char \* > **cmatch**
- typedef [regex\\_iterator](#)< const  
    char \* > **cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)  
    < const char \* > [cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [csub\\_match](#)
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- typedef [integral\\_constant](#)  
    < bool, false > [false\\_type](#)
- typedef [basic\\_filebuf](#)< char > [filebuf](#)
- typedef [basic\\_fstream](#)< char > [fstream](#)

- typedef [basic\\_ifstream](#)< char > [ifstream](#)
- typedef [basic\\_ios](#)< char > [ios](#)
- typedef [basic\\_iostream](#)< char > [iostream](#)
- typedef [basic\\_istream](#)< char > [istream](#)
- typedef [basic\\_istreamstream](#)< char > [istreamstream](#)
- typedef [shuffle\\_order\\_engine](#)  
< [minstd\\_rand0](#), 256 > [knuth\\_b](#)
- typedef enum [std::memory\\_order](#) [memory\\_order](#)
- typedef  
[linear\\_congruential\\_engine](#)  
< [uint\\_fast32\\_t](#), 48271UL, 0UL, 2147483647UL > [minstd\\_rand](#)
- typedef  
[linear\\_congruential\\_engine](#)  
< [uint\\_fast32\\_t](#), 16807UL, 0UL, 2147483647UL > [minstd\\_rand0](#)
- typedef  
[mersenne\\_twister\\_engine](#)  
< [uint\\_fast32\\_t](#), 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef  
[mersenne\\_twister\\_engine](#)  
< [uint\\_fast64\\_t](#), 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000-ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > [mt19937\\_64](#)
- typedef void(\* [new\\_handler](#) )()
- typedef [basic\\_ofstream](#)< char > [ofstream](#)
- typedef [basic\\_ostream](#)< char > [ostream](#)
- typedef [basic\\_ostreamstream](#)< char > [ostreamstream](#)
- typedef [\\_\\_PTRDIFF\\_TYPE\\_\\_](#) [ptrdiff\\_t](#)
- typedef [discard\\_block\\_engine](#)  
< [ranlux24\\_base](#), 223, 23 > [ranlux24](#)
- typedef  
[subtract\\_with\\_carry\\_engine](#)  
< [uint\\_fast32\\_t](#), 24, 10, 24 > [ranlux24\\_base](#)
- typedef [discard\\_block\\_engine](#)  
< [ranlux48\\_base](#), 389, 11 > [ranlux48](#)
- typedef  
[subtract\\_with\\_carry\\_engine](#)  
< [uint\\_fast64\\_t](#), 48, 5, 12 > [ranlux48\\_base](#)
- template<typename [\\_R1](#) , typename [\\_R2](#) >  
using [ratio\\_divide](#) = typename [\\_\\_ratio\\_divide](#)< [\\_R1](#), [\\_R2](#) >::type
- template<typename [\\_R1](#) , typename [\\_R2](#) >  
using [ratio\\_multiply](#) = typename [\\_\\_ratio\\_multiply](#)< [\\_R1](#), [\\_R2](#) >::type
- typedef [basic\\_regex](#)< char > [regex](#)
- typedef [\\_\\_SIZE\\_TYPE\\_\\_](#) [size\\_t](#)
- typedef [match\\_results](#)  
< [string::const\\_iterator](#) > [smatch](#)
- typedef [regex\\_iterator](#)  
< [string::const\\_iterator](#) > [sregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)  
< [string::const\\_iterator](#) > [sregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)  
< [string::const\\_iterator](#) > [ssub\\_match](#)
- typedef [basic\\_streambuf](#)< char > [streambuf](#)

- typedef long long [streamoff](#)
- typedef [fpos](#)< mbstate\_t > [streampos](#)
- typedef ptrdiff\_t [streamsize](#)
- typedef [basic\\_string](#)< char > [string](#)
- typedef [basic\\_stringbuf](#)< char > [stringbuf](#)
- typedef [basic\\_stringstream](#)< char > [stringstream](#)
- typedef void(\* [terminate\\_handler](#) )()
- typedef [integral\\_constant](#)  
    < bool, true > [true\\_type](#)
- typedef [fpos](#)< mbstate\_t > [u16streampos](#)
- typedef [basic\\_string](#)< char16\_t > [u16string](#)
- typedef [fpos](#)< mbstate\_t > [u32streampos](#)
- typedef [basic\\_string](#)< char32\_t > [u32string](#)
- typedef void(\* [unexpected\\_handler](#) )()
- typedef [match\\_results](#)< const  
    wchar\_t \* > **wcmatch**
- typedef [regex\\_iterator](#)< const  
    wchar\_t \* > **wcregex\_iterator**
- typedef [regex\\_token\\_iterator](#)  
    < const wchar\_t \* > [wcregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const  
    wchar\_t \* > [wsub\\_match](#)
- typedef [basic\\_filebuf](#)< wchar\_t > [wfilebuf](#)
- typedef [basic\\_fstream](#)< wchar\_t > [wfstream](#)
- typedef [basic\\_ifstream](#)< wchar\_t > [wifstream](#)
- typedef [basic\\_ios](#)< wchar\_t > [wios](#)
- typedef [basic\\_iostream](#)< wchar\_t > [wiostream](#)
- typedef [basic\\_istream](#)< wchar\_t > [wistream](#)
- typedef [basic\\_istream](#)  
    < wchar\_t > [wistream](#)
- typedef [basic\\_ofstream](#)< wchar\_t > [wofstream](#)
- typedef [basic\\_ostream](#)< wchar\_t > [wostream](#)
- typedef [basic\\_ostream](#)  
    < wchar\_t > [wostream](#)
- typedef [basic\\_regex](#)< wchar\_t > [wregex](#)
- typedef [match\\_results](#)  
    < wstring::const\_iterator > **wsmatch**
- typedef [regex\\_iterator](#)  
    < wstring::const\_iterator > **wsregex\_iterator**
- typedef [regex\\_token\\_iterator](#)  
    < wstring::const\_iterator > [wsregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)  
    < wstring::const\_iterator > [wsub\\_match](#)
- typedef [basic\\_streambuf](#)< wchar\_t > [wstreambuf](#)
- typedef [fpos](#)< mbstate\_t > [wstreampos](#)
- typedef [basic\\_string](#)< wchar\_t > [wstring](#)
- typedef [basic\\_stringbuf](#)< wchar\_t > [wstringbuf](#)
- typedef [basic\\_stringstream](#)  
    < wchar\_t > [wstringstream](#)

## Enumerations

- enum { [\\_S\\_threshold](#) }
- enum { [\\_S\\_chunk\\_size](#) }
- enum { [\\_S\\_word\\_bit](#) }
- enum [\\_\\_memory\\_order\\_modifier](#) { [\\_\\_memory\\_order\\_mask](#), [\\_\\_memory\\_order\\_modifier\\_mask](#), [\\_\\_memory\\_order\\_hle\\_acquire](#), [\\_\\_memory\\_order\\_hle\\_release](#) }
- enum [\\_ios\\_Fmtflags](#) { [\\_S\\_boolalpha](#), [\\_S\\_dec](#), [\\_S\\_fixed](#), [\\_S\\_hex](#), [\\_S\\_internal](#), [\\_S\\_left](#), [\\_S\\_oct](#), [\\_S\\_right](#), [\\_S\\_scientific](#), [\\_S\\_showbase](#), [\\_S\\_showpoint](#), [\\_S\\_showpos](#), [\\_S\\_skipws](#), [\\_S\\_unitbuf](#), [\\_S\\_uppercase](#), [\\_S\\_adjustfield](#), [\\_S\\_basefield](#), [\\_S\\_floatfield](#), [\\_S\\_ios\\_fmtflags\\_end](#) }
- enum [\\_ios\\_iostate](#) { [\\_S\\_goodbit](#), [\\_S\\_badbit](#), [\\_S\\_eofbit](#), [\\_S\\_failbit](#), [\\_S\\_ios\\_iostate\\_end](#) }
- enum [\\_ios\\_Openmode](#) { [\\_S\\_app](#), [\\_S\\_ate](#), [\\_S\\_bin](#), [\\_S\\_in](#), [\\_S\\_out](#), [\\_S\\_trunc](#), [\\_S\\_ios\\_openmode\\_end](#) }
- enum [\\_ios\\_Seekdir](#) { [\\_S\\_beg](#), [\\_S\\_cur](#), [\\_S\\_end](#), [\\_S\\_ios\\_seekdir\\_end](#) }
- enum [\\_\\_Manager\\_operation](#) { [\\_\\_get\\_type\\_info](#), [\\_\\_get\\_functor\\_ptr](#), [\\_\\_clone\\_functor](#), [\\_\\_destroy\\_functor](#) }
- enum [\\_Rb\\_tree\\_color](#) { [\\_S\\_red](#), [\\_S\\_black](#) }
- enum [cv\\_status](#)
- enum [errc](#)
- enum [float\\_denorm\\_style](#) { [denorm\\_indeterminate](#), [denorm\\_absent](#), [denorm\\_present](#) }
- enum [float\\_round\\_style](#) { [round\\_indeterminate](#), [round\\_toward\\_zero](#), [round\\_to\\_nearest](#), [round\\_toward\\_infinity](#), [round\\_toward\\_neg\\_infinity](#) }
- enum [future\\_errc](#)
- enum [future\\_status](#)
- enum [launch](#)
- enum [memory\\_order](#) { [memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#), [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

## Functions

- template<typename [\\_CharT](#) >  
[\\_CharT \\* \\_\\_add\\_grouping](#) ([\\_CharT \\* \\_\\_s](#), [\\_CharT \\_\\_sep](#), const char \* [\\_\\_gbeg](#), size\_t [\\_\\_gsize](#), const [\\_CharT \\* \\_\\_first](#), const [\\_CharT \\* \\_\\_last](#))
- template<typename [\\_Tp](#) >  
[\\_Tp \\* \\_\\_addressof](#) ([\\_Tp &\\_\\_r](#)) noexcept
- template<typename [\\_RandomAccessIterator](#), typename [\\_Distance](#), typename [\\_Tp](#) >  
void [\\_\\_adjust\\_heap](#) ([\\_RandomAccessIterator \\_\\_first](#), [\\_Distance \\_\\_holeIndex](#), [\\_Distance \\_\\_len](#), [\\_Tp \\_\\_value](#))
- template<typename [\\_RandomAccessIterator](#), typename [\\_Distance](#), typename [\\_Tp](#), typename [\\_Compare](#) >  
void [\\_\\_adjust\\_heap](#) ([\\_RandomAccessIterator \\_\\_first](#), [\\_Distance \\_\\_holeIndex](#), [\\_Distance \\_\\_len](#), [\\_Tp \\_\\_value](#), [\\_Compare \\_\\_comp](#))
- template<typename [\\_InputIterator](#), typename [\\_Distance](#) >  
void [\\_\\_advance](#) ([\\_InputIterator &\\_\\_i](#), [\\_Distance \\_\\_n](#), [input\\_iterator\\_tag](#))
- template<typename [\\_BidirectionalIterator](#), typename [\\_Distance](#) >  
void [\\_\\_advance](#) ([\\_BidirectionalIterator &\\_\\_i](#), [\\_Distance \\_\\_n](#), [bidirectional\\_iterator\\_tag](#))

- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __advance (_RandomAccessIterator &__i, _Distance __n, random\_access\_iterator\_tag)`
- `template<typename _Alloc >`  
`void __alloc_on_copy (_Alloc &__one, const _Alloc &__two)`
- `template<typename _Alloc >`  
`_Alloc __alloc_on_copy (const _Alloc &__a)`
- `template<typename _Alloc >`  
`void __alloc_on_move (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc >`  
`void __alloc_on_swap (_Alloc &__one, _Alloc &__two)`
- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > __allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Callable, typename... _Args>`  
`_Bind_simple_helper< _Callable,`  
`_Args...>::type __bind_simple (_Callable &&__callable, _Args &&... __args)`
- `template<typename _Functor >`  
`_Functor & __callable_functor (_Functor &__f)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _Class::*&__p)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _Class::*const &__p)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _Class::*volatile &__p)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _Class::*const volatile &__p)`
- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet *__f)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __-`  
`chunk_size)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __-`  
`chunk_size, _Compare __comp)`
- `constexpr memory\_order __cmpexch_failure_order (memory\_order __m) noexcept`
- `constexpr memory\_order __cmpexch_failure_order2 (memory\_order __m) noexcept`
- `template<typename _Tp >`  
`_Tp __complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`

- `template<typename _Tp >`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc __attribute__((__unused__)), char * __out, const int __size __attribute__((__unused__)), const char * __fmt,...)`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`ostreambuf_iterator< _CharT >`  
`>::__type __copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`ostreambuf_iterator< _CharT >`  
`>::__type __copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`_CharT * >::__type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT >`  
`__last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`ostreambuf_iterator< _CharT,`  
`char_traits< _CharT >`  
`>::__type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT,`  
`char_traits< _CharT > >)`

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`ostreambuf_iterator< _CharT,`  
`char_traits< _CharT >`  
`> >::__type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits<`  
`_CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`_CharT * >::__type __copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_`  
`iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access-`  
`iterator_tag)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _`  
`Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits`  
`> *, bool &)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sbout,`  
`bool & __ineof)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__`  
`_sbout, bool & __ineof)`
- `size_t __deque_buf_size (size_t __size)`
- `template<typename _InputIterator >`  
`iterator_traits`  
`< _InputIterator >`  
`::difference_type __distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`iterator_traits`  
`< _RandomAccessIterator >`  
`::difference_type __distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access-`  
`iterator_tag)`
- `template<typename _Alloc >`  
`void __do_alloc_on_copy (_Alloc & __one, const _Alloc & __two, true_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_move (_Alloc & __one, _Alloc & __two, true_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_swap (_Alloc & __one, _Alloc & __two, true_type)`
- `template<typename _Alloc >`  
`auto __do_outermost (_Alloc & __a, _Alloc *) -> decltype(__a.outer_allocator())`

- `template<typename _Alloc >`  
`_Alloc & __do_outmost (_Alloc &__a,...)`
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >`  
`void __enable_shared_from_this_helper (const __shared_count< _Lp > &, const __enable_shared_from_this< _Tp1, _Lp > *, const _Tp2 *) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`void __enable_shared_from_this_helper (const __shared_count<> &, const enable\_shared\_from\_this< _Tp1 > *, const _Tp2 *) noexcept`
- `template<_Lock_policy _Lp>`  
`void __enable_shared_from_this_helper (const __shared_count< _Lp > &,...) noexcept`
- `template<typename _I1, typename _I2 >`  
`bool __equal_aux (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`\_\_gnu\_cxx::\_\_enable\_if  
<!__is_scalar< _Tp >>::__value,  
void >::__type __fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`  
`\_\_gnu\_cxx::\_\_enable\_if  
<__is_byte< _Tp >>::__value,  
void >::__type __fill_a (_Tp *__first, _Tp *__last, const _Tp &__c)`
- `void __fill_bvector (_Bit_iterator __first, _Bit_iterator __last, bool __x)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`\_\_gnu\_cxx::\_\_enable\_if  
<!__is_scalar< _Tp >>::__value,  
_OutputIterator >::__type __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`  
`\_\_gnu\_cxx::\_\_enable\_if  
<__is_byte< _Tp >>::__value,  
_Tp * >::__type __fill_n_a (_Tp *__first, _Size __n, const _Tp &__c)`
- `template<typename _RandomAccessIterator >`  
`void __final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator __find (_InputIterator __first, _InputIterator __last, const _Tp &__val, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator __find (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_iterator\_tag, forward\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`  
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >`  
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input\_iterator\_tag)`

- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator __find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate -`  
`__pred, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator __find_if_not (_RandomAccessIterator __first, _RandomAccessIterator __last, _`  
`Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance >`  
`_InputIterator __find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)`
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement __gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr __add_ref< _Head >::type __get_helper (\_Tuple\_impl< __i, _Head, _Tail...> &__t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr __add_c_ref< _Head >`  
`::type __get_helper (const \_Tuple\_impl< __i, _Head, _Tail...> &__t) noexcept`
- `template<typename _Ex >`  
`const nested\_exception * __get_nested_exception (const _Ex &__ex)`
- `mutex & __get_once_mutex ()`
- `template<typename _RandomAccessIterator >`  
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _Tp >`  
`size_t __iconv_adapter (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf,`  
`size_t *__inbytes, char **__outbuf, size_t *__outbytes)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`  
`_ForwardIterator __inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`  
`void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT >`  
`int __int_to_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios\_base::fmtflags __flags, bool __dec)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void __introslect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __`  
`last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void __introslect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __`  
`last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`

- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _Functor, typename... _Args>`  
`enable_if< (lis_member_pointer`  
`< _Functor >::value`  
`&&lis_function< _Functor >`  
`::value &&lis_function`  
`< typename remove_pointer`  
`< _Functor >::type >::value),`  
`typename result_of< _Functor(_Args &&...)>`  
`::type >::type __invoke (_Functor &__f, _Args &&...__args)`
- `template<typename _Functor, typename... _Args>`  
`enable_if< (is_pointer`  
`< _Functor >::value`  
`&&is_function< typename`  
`remove_pointer< _Functor >`  
`::type >::value), typename`  
`result_of< _Functor(_Args &&...)>`  
`::type >::type __invoke (_Functor __f, _Args &&...__args)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool __is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _Iter >`  
`iterator_traits< _Iter >`  
`::iterator_category __iterator_category (const _Iter &)`
- `template<typename _I1, typename _I2 >`  
`bool __lexicographical_compare_aux (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `constexpr int __lg (int __n)`
- `constexpr unsigned __lg (unsigned __n)`
- `constexpr long __lg (long __n)`
- `constexpr unsigned long __lg (unsigned long __n)`
- `constexpr long long __lg (long long __n)`
- `constexpr unsigned long long __lg (unsigned long long __n)`
- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`  
`_ReturnType __make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > __make_shared (_Args &&...__args)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`  
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`

- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`  
`void \_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last,`  
`_Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance >`  
`void \_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccess-`  
`Iterator2 __result, _Distance __step_size)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`  
`void \_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccess-`  
`Iterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer >`  
`void \_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __-`  
`buffer)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`  
`void \_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __-`  
`buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void \_\_merge\_without\_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator`  
`__last, _Distance __len1, _Distance __len2)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`  
`void \_\_merge\_without\_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator`  
`__last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator >`  
`_Miter_base< _Iterator >`  
`::iterator_type \_\_miter\_base (_Iterator __it)`
- `template<typename _Iterator >`  
`void \_\_move\_median\_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _Iterator, typename _Compare >`  
`void \_\_move\_median\_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator \_\_move\_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`  
`Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_move\_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`  
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`void \_\_move\_merge\_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`  
`Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`void \_\_move\_merge\_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`  
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3 >`  
`void \_\_move\_merge\_adaptive\_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _-`  
`BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`  
`void \_\_move\_merge\_adaptive\_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _-`  
`BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >`  
`::iterator_type \_\_niter\_base (_Iterator __it)`
- `void \_\_once\_proxy (void)`
- `template<typename _CharT, typename _Traits >`  
`void \_\_ostream\_fill (basic\_ostream< _CharT, _Traits > &__out, streamsize __n)`

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _`  
`CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _Alloc >`  
`auto __outermost (_Alloc &__a) -> decltype(__do_outermost(__a,&__a))`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_`  
`iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred,`  
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __`  
`result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __`  
`result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value,`  
`_Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`void __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last,`  
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __`  
`last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`  
`_BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _`  
`BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance`  
`__buffer_size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &_`  
`__val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp >`  
`_RandomAccessIter __search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count,`  
`const _Tp &__val, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &_`  
`__val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RandomAccessIter __search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count,`  
`const _Tp &__val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)`
- `void __set_once_functor_lock_ptr (unique_lock< mutex > *)`

- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`  
`_ForwardIterator \_\_stable\_partition\_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance >`  
`void \_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`  
`void \_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `void \_\_throw\_bad\_alloc (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_cast (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_exception (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_function\_call () __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_typeid (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_weak\_ptr ()`
- `void \_\_throw\_domain\_error (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_future\_error (int) __attribute__((__noreturn__))`
- `void \_\_throw\_invalid\_argument (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_ios\_failure (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_length\_error (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_logic\_error (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_out\_of\_range (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_overflow\_error (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_range\_error (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_regex\_error (regex\_constants::error\_type __ecode)`
- `void \_\_throw\_runtime\_error (const char *) __attribute__((__noreturn__))`
- `void \_\_throw\_system\_error (int) __attribute__((__noreturn__))`
- `void \_\_throw\_underflow\_error (const char *) __attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void \_\_throw\_with\_nested (_Ex &&, const nested\_exception *e) __attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void \_\_throw\_with\_nested (_Ex &&, ...) __attribute__((__noreturn__))`
- `template<typename _Lock >`  
`unique\_lock<_Lock> \_\_try\_to\_lock (_Lock &__l)`
- `template<typename _RandomAccessIterator >`  
`void \_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void \_\_unguarded\_linear\_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator \_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`  
`_RandomAccessIterator \_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator \_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator \_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Pointer, typename _ForwardIterator >`  
`void \_\_uninitialized\_construct\_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_copy\_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator \_\_uninitialized\_copy\_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator<_Tp> &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_copy\_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator \_\_uninitialized\_copy\_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator \_\_uninitialized\_copy\_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void \_\_uninitialized\_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void \_\_uninitialized\_default\_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void \_\_uninitialized\_default\_a (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &)`
- `template<typename _ForwardIterator, typename _Size >`  
`void \_\_uninitialized\_default\_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`  
`void \_\_uninitialized\_default\_n\_a (_ForwardIterator __first, _Size __n, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`void \_\_uninitialized\_default\_n\_a (_ForwardIterator __first, _Size __n, allocator<_Tp> &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void \_\_uninitialized\_fill\_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`  
`void \_\_uninitialized\_fill\_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x, allocator<_Tp2> &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_fill\_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp & __x, _InputIterator __first, _InputIterator __last, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`  
`void \_\_uninitialized\_fill\_n\_a (_ForwardIterator __first, _Size __n, const _Tp & __x, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`  
`void \_\_uninitialized\_fill\_n\_a (_ForwardIterator __first, _Size __n, const _Tp & __x, allocator<_Tp2> &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_move\_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_move\_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator & __alloc)`

- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _Bi_iter >`  
`const sub_match< _Bi_iter > & __unmatched_sub ()`
- `template<typename _Tp, typename _Alloc, typename... _Args >`  
`__uses_alloc_impl< _Tp, _Alloc, _Args...> __use_alloc (const _Alloc &__a)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`

- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array<_Tp> __a, _Array<size_t> __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void *__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict __valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`
- `template<std::size_t _Ind, typename... _Tp>`  
`auto __volget (volatile tuple<_Tp...> &__tuple) -> typename tuple_element<_Ind, tuple<_Tp...>>::type volatile &`
- `template<std::size_t _Ind, typename... _Tp>`  
`auto __volget (const volatile tuple<_Tp...> &__tuple) -> typename tuple_element<_Ind, tuple<_Tp...>>::type const volatile &`
- `template<typename _CharT >`  
`ostreambuf_iterator<_CharT> __write (ostreambuf_iterator<_CharT> __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _OutIter >`  
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, _Array<size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _T1, typename... _Args>`  
`void _Construct (_T1 *__p, _Args &&...__args)`
- `template<typename _Tp >`  
`void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`
- `size_t _Find_first () const noexcept`
- `size_t _Find_next (size_t __prev) const noexcept`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `template<class _CharT, class _Traits >`  
`void _M_copy_from_ptr (const _CharT *, size_t, size_t, size_t, _CharT, _CharT)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void _M_copy_from_string (const std::basic\_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void _M_copy_from_string (const std::basic\_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void _M_copy_to_string (std::basic\_string< _CharT, _Traits, _Alloc > &, _CharT, _CharT) const`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void _M_copy_to_string (std::basic\_string< _CharT, _Traits, _Alloc > &__s) const`
- `template<size_t _Nb>`  
`_M_do_and (__rhs)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()`

- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_decrement (const _Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_increment (const _Rb_tree_node_base *__x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `void abort (void) throw ()`
- `template<typename _Tp >  
_Tp abs (const complex< _Tp > &)`
- `constexpr double abs (double __x)`
- `constexpr float abs (float __x)`
- `constexpr long double abs (long double __x)`
- `template<typename _Tp >  
constexpr  
__gnu_cxx::__enable_if  
< __is_integer< _Tp >::__value,  
double >::__type abs (_Tp __x)`
- `template<class _Dom >  
_Expr< _UnClos< _Abs, _Expr,  
_Dom >, typename  
_Dom::value_type > abs (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Abs,  
_ValArray, _Tp >, _Tp > abs (const valarray< _Tp > &__v)`
- `template<typename _InputIterator, typename _Tp >  
_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >  
_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `constexpr float acos (float __x)`
- `constexpr long double acos (long double __x)`
- `template<typename _Tp >  
constexpr  
__gnu_cxx::__enable_if  
< __is_integer< _Tp >::__value,  
double >::__type acos (_Tp __x)`
- `template<class _Dom >  
_Expr< _UnClos< _Acos, _Expr,  
_Dom >, typename  
_Dom::value_type > acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Acos,  
_ValArray, _Tp >, _Tp > acos (const valarray< _Tp > &__v)`
- `template<typename _Tp >  
std::complex< _Tp > acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< _Tp > acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
_Tp * addressof (_Tp &__r) noexcept`
- `template<typename _InputIterator, typename _OutputIterator >  
_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter >`  
`_Filter adjacent\_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter adjacent\_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance >`  
`void advance (_InputIterator &__i, _Distance __n)`
- `bool all () const noexcept`
- `template<typename _Iter, typename _Predicate >`  
`bool all\_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared\_ptr<_Tp> allocate\_shared (const _Alloc &__a, _Args &&... __args)`
- `bool any () const noexcept`
- `template<typename _Iter, typename _Predicate >`  
`bool any\_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp arg (const complex<_Tp> &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote<_Tp>::__type arg (_Tp __x)`
- `constexpr float asin (float __x)`
- `constexpr long double asin (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer<_Tp>::__value,`  
`double >::__type asin (_Tp __x)`
- `template<class _Dom >`  
`_Expr<_UnClos<_Asin, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > asin (const _Expr<_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr<_UnClos<_Asin,`  
`_ValArray, _Tp >, _Tp > asin (const valarray<_Tp> &__v)`
- `template<typename _Tp >`  
`std::complex<_Tp> asin (const std::complex<_Tp> &__z)`
- `template<typename _Tp >`  
`std::complex<_Tp> asinh (const std::complex<_Tp> &__z)`
- `template<typename _Fn, typename... _Args>`  
`future< typename result\_of`  
`<_Fn(_Args...)>::__type > async (launch __policy, _Fn &&__fn, _Args &&... __args)`

- `template<typename _Fn, typename... _Args>`  
`future< typename result\_of`  
`< _Fn(_Args...)>::type > async (_Fn &&__fn, _Args &&...__args)`
- `constexpr float atan (float __x)`
- `constexpr long double atan (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type atan (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Atan, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Atan,`  
`_ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`std::complex< _Tp > atan (const std::complex< _Tp > &__z)`
- `constexpr float atan2 (float __y, float __x)`
- `constexpr long double atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`  
`constexpr`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type atan2 (_Tp __y, _Up __x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, _Tp > atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`_Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Atan2, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename _Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr,`  
`_ValArray, _Dom, typename`  
`_Dom::value_type >, typename`  
`_Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _Dom::value_type > atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`

- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, _Tp > atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, _Tp > atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _Dom::value_type > atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<typename _Tp >`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &__z)`
- `int atexit (void(*)()) throw ()`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory\_order`  
`__m1, memory\_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2,`  
`memory\_order __m1, memory\_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory\_order`  
`__m1, memory\_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2,`  
`memory\_order __m1, memory\_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (atomic< _ITp > *__a, _ITp __i, memory\_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory\_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (\_\_atomic\_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (volatile \_\_atomic\_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`

- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `void atomic_flag_clear (atomic_flag *__a) noexcept`

- void **atomic\_flag\_clear** (volatile [atomic\\_flag](#) \*\_\_a) noexcept
- void **atomic\_flag\_clear\_explicit** ([atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- void **atomic\_flag\_clear\_explicit** (volatile [atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- bool **atomic\_flag\_test\_and\_set** ([atomic\\_flag](#) \*\_\_a) noexcept
- bool **atomic\_flag\_test\_and\_set** (volatile [atomic\\_flag](#) \*\_\_a) noexcept
- bool **atomic\_flag\_test\_and\_set\_explicit** ([atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- bool **atomic\_flag\_test\_and\_set\_explicit** (volatile [atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
void **atomic\_init** ([atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
void **atomic\_init** (volatile [atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
bool **atomic\_is\_lock\_free** (const [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
bool **atomic\_is\_lock\_free** (const volatile [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
\_ITp **atomic\_load** (const [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
\_ITp **atomic\_load** (const volatile [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
\_ITp **atomic\_load\_explicit** (const [atomic](#)< \_ITp > \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
\_ITp **atomic\_load\_explicit** (const volatile [atomic](#)< \_ITp > \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- void **atomic\_signal\_fence** ([memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
void **atomic\_store** ([atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
void **atomic\_store** (volatile [atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
void **atomic\_store\_explicit** ([atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
void **atomic\_store\_explicit** (volatile [atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m) noexcept
- void **atomic\_thread\_fence** ([memory\\_order](#) \_\_m) noexcept
- template<typename \_Container >  
[back\\_insert\\_iterator](#)< \_Container > **back\_inserter** (\_Container &\_\_x)
- template<class \_Container >  
auto **begin** (\_Container &\_\_cont) -> decltype(\_\_cont.begin())
- template<class \_Container >  
auto **begin** (const \_Container &\_\_cont) -> decltype(\_\_cont.begin())
- template<class \_Tp, size\_t \_Nm >  
\_Tp \* **begin** (\_Tp(&\_\_arr)[\_Nm])
- template<class \_Tp >  
constexpr const \_Tp \* **begin** ([initializer\\_list](#)< \_Tp > \_\_ils) noexcept
- template<class \_Tp >  
\_Tp \* **begin** ([valarray](#)< \_Tp > &\_\_va)
- template<class \_Tp >  
const \_Tp \* **begin** (const [valarray](#)< \_Tp > &\_\_va)
- template<typename \_Filter, typename \_Tp >  
bool **binary\_search** (\_Filter, \_Filter, const \_Tp &)
- template<typename \_Filter, typename \_Tp, typename \_Compare >  
bool **binary\_search** (\_Filter, \_Filter, const \_Tp &, \_Compare)

- `template<typename _ForwardIterator, typename _Tp >`  
`bool binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Func, typename... _BoundArgs>`  
`_Bind_helper< __is_socketlike`  
`< _Func >::value, _Func,`  
`_BoundArgs...>::type bind (_Func &&__f, _BoundArgs &&...__args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`_Bindres_helper< _Result,`  
`_Func, _BoundArgs...>::type bind (_Func &&__f, _BoundArgs &&...__args)`
- `template<typename _Operation, typename _Tp >`  
`binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp >`  
`binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)`
- `ios\_base & boolalpha (ios\_base &__base)`
- `template<typename _Callable, typename... _Args>`  
`void call\_once (once\_flag &__once, _Callable &&__f, _Args &&...__args)`
- `constexpr float ceil (float __x)`
- `constexpr long double ceil (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type ceil (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type conj (_Tp __x)`
- `template<typename _Tp, typename _Tp1 >`  
`shared\_ptr< _Tp > const\_pointer\_cast (const shared\_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > const\_pointer\_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Iter, typename _OIter >`  
`_OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`ostreambuf\_iterator< _CharT >`  
`>::__type copy (istreambuf\_iterator< _CharT > __first, istreambuf\_iterator< _CharT > __last, ostreambuf\_`  
`iterator< _CharT > __result)`
- `template<typename _Tp >`  
`Deque\_iterator< _Tp, _Tp`  
`&, _Tp * > copy (Deque\_iterator< _Tp, _Tp &, _Tp * > __first, Deque\_iterator< _Tp, _Tp &, _Tp * > __last,`  
`Deque\_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI >`  
`_OI copy (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`  
`Deque\_iterator< _Tp, _Tp`  
`&, _Tp * > copy (Deque\_iterator< _Tp, const _Tp &, const _Tp * > __first, Deque\_iterator< _Tp, const _Tp`  
`&, const _Tp * > __last, Deque\_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Blter1, typename _Blter2 >`  
`_Blter2 copy\_backward (_Blter1, _Blter1, _Blter2)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > copy_backward ( _Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp`  
`* > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1 , typename _BI2 >`  
`_BI2 copy_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > copy_backward ( _Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Ex >`  
`exception_ptr copy_exception ( _Ex __ex) noexcept`
- `template<typename _Iter , typename _OIter , typename _Predicate >`  
`_OIter copy_if ( _Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate >`  
`_OutputIterator copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter , typename _Size , typename _OIter >`  
`_OIter copy_n ( _Iter, _Size, _OIter)`
- `template<typename _InputIterator , typename _Size , typename _OutputIterator >`  
`_OutputIterator copy_n ( _InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _Tp >`  
`complex< _Tp > cos (const complex< _Tp > &)`
- `constexpr float cos (float __x)`
- `constexpr long double cos (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type cos ( _Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cos, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > cos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cos,`  
`_ValArray, _Tp >, _Tp > cos (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`complex< _Tp > cosh (const complex< _Tp > &)`
- `constexpr float cosh (float __x)`
- `constexpr long double cosh (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type cosh ( _Tp __x)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cosh,`  
`_ValArray, _Tp >, _Tp > cosh (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cosh, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type count ( _Iter, _Iter, const _Tp &)`
- `size_t count () const noexcept`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits`  
`< _InputIterator >`  
`::difference_type count ( _InputIterator __first, _InputIterator __last, const _Tp & __value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type count_if ( _Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits`  
`< _InputIterator >`  
`::difference_type count_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `exception_ptr current_exception () noexcept`
- `ios_base & dec (ios_base & __base)`
- `template<typename _Tp >`  
`add_rvalue_reference< _Tp >::type declval () noexcept`
- `template<typename _InputIterator >`  
`iterator_traits`  
`< _InputIterator >`  
`::difference_type distance ( _InputIterator __first, _InputIterator __last)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Tp1 > & __r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp >`  
`__shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > & __r) noexcept`
- `template<class _Container >`  
`auto end ( _Container & __cont) -> decltype(__cont.end())`
- `template<class _Container >`  
`auto end (const _Container & __cont) -> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm >`  
`_Tp * end ( _Tp(& __arr)[_Nm])`
- `template<class _Tp >`  
`constexpr const _Tp * end (initializer_list< _Tp > __ils) noexcept`
- `template<class _Tp >`  
`_Tp * end (valarray< _Tp > & __va)`
- `template<class _Tp >`  
`const _Tp * end (const valarray< _Tp > & __va)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits > & __os)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > & __os)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal ( _Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal ( _Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2 >`  
`bool equal ( _I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > equal_range ( _Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > equal_range ( _Filter, _Filter, const _Tp &, _Compare)`

- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator,`  
`_ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator,`  
`_ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare`  
`__comp)`
- `void exit (int) throw ()`
- `template<typename _Tp >`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `constexpr float exp (float __x)`
- `constexpr long double exp (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type exp (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Exp, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Exp,`  
`_ValArray, _Tp >, _Tp > exp (const valarray< _Tp > &__v)`
- `constexpr float fabs (float __x)`
- `constexpr long double fabs (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type fabs (_Tp __x)`
- `template<typename _Tp >`  
`_Tp fabs (const std::complex< _Tp > &__z)`
- `template<typename _Filter, typename _Tp >`  
`void fill (_Filter, _Filter, const _Tp &)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`  
`void fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp * >`  
`&__last, const _Tp &__value)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`istreambuf_iterator< _CharT >`  
`>::__type find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT`  
`&__val)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter, _Iter, const _Tp &)`

- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 find\_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 find\_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 find\_first\_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 find\_first\_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find\_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find\_if\_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios\_base & fixed (ios\_base &__base)`
- `bitset< _Nb > & flip () noexcept`
- `bitset< _Nb > & flip (size_t __position)`
- `constexpr float floor (float __x)`
- `constexpr long double floor (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type floor (_Tp __x)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & flush (basic\_ostream< _CharT, _Traits > &__os)`
- `constexpr float fmod (float __x, float __y)`
- `constexpr long double fmod (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`constexpr`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type fmod (_Tp __x, _Up __y)`
- `template<typename _Iter, typename _Funct >`  
`_Funct for\_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`  
`_Function for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`

- `template<typename _Tp >`  
`constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &&__t) noexcept`
- `template<typename... _Elements>`  
`tuple< _Elements &&...> forward_as_tuple ( _Elements &&... __args) noexcept`
- `float frexp (float __x, int * __exp)`
- `long double frexp (long double __x, int * __exp)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type frexp ( _Tp __x, int * __exp)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > front_inserter ( _Container &__x)`
- `const error_category & future_category () noexcept`
- `template<typename _Filter, typename _Generator >`  
`void generate ( _Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void generate ( _ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`  
`_RealType generate_canonical ( _UniformRandomNumberGenerator &__g)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n ( _OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n ( _OutputIterator __first, _Size __n, _Generator __gen)`
- `const error_category & generic_category () noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr tuple_element< _Int,`  
`std::pair< _Tp1, _Tp2 >`  
`>::type & get (std::pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr tuple_element< _Int,`  
`std::pair< _Tp1, _Tp2 >`  
`>::type && get (std::pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr const tuple_element`  
`< _Int, std::pair< _Tp1, _Tp2 >`  
`>::type & get (const std::pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp & get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp && get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp & get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __add_ref< typename`  
`tuple_element< __i, tuple`  
`< _Elements...> >::type >`  
`::type get (tuple< _Elements...> &__t) noexcept`

- `template<std::size_t __i, typename... _Elements>`  
`constexpr __add_c_ref`  
`< typename tuple_element< __i,`  
`tuple< _Elements...> >::type >`  
`::type get (const tuple< _Elements...> &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __add_r_ref`  
`< typename tuple_element< __i,`  
`tuple< _Elements...> >::type >`  
`::type get (tuple< _Elements...> &&__t) noexcept`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _Tp >`  
`pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_`  
`string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_`  
`string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT,`  
`_Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT,`  
`_Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<>`  
`basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_`  
`t __delim)`
- `template<typename _Facet >`  
`bool has_facet (const locale &__loc) throw ()`
- `ios_base & hex (ios_base &__base)`
- `template<typename _Tp >`  
`constexpr _Tp imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type imag (_Tp)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`  
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`  
`_Tp inner_product ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _Binary-`  
`Operation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _BIter >`  
`void inplace_merge ( _BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void inplace_merge ( _BIter, _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`void inplace_merge ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void inplace_merge ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _-`  
`Compare __comp)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > inserter ( _Container &__x, _Iterator __i)`
- `ios_base & internal (ios_base &__base)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void iota ( _ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter >`  
`bool is_heap ( _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool is_heap ( _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`bool is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`_RAIter is_heap_until ( _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter is_heap_until ( _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator is_heap_until ( _RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator is_heap_until ( _RandomAccessIterator __first, _RandomAccessIterator __last, _-`  
`Compare __comp)`
- `template<typename _Iter, typename _Predicate >`  
`bool is_partitioned ( _Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool is_partitioned ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter1, typename _Filter2 >`  
`bool is_permutation ( _Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`bool is_permutation ( _Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _Binary-`  
`Predicate __pred)`
- `template<typename _Filter >`  
`bool is_sorted ( _Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`bool is_sorted ( _Filter, _Filter, _Compare)`

- `template<typename _ForwardIterator >`  
`bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Filter >`  
`_Filter is\_sorted\_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter is\_sorted\_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT >`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void iter\_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Filter1, typename _Filter2 >`  
`void iter\_swap (_Filter1, _Filter2)`
- `template<typename _Tp >`  
`_Tp kill\_dependency (_Tp __y) noexcept`
- `constexpr float ldexp (float __x, int __exp)`
- `constexpr long double ldexp (long double __x, int __exp)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`<__is_integer< _Tp >::__value,`  
`double >::__type ldexp (_Tp __x, int __exp)`
- `ios\_base & left (ios\_base &__base)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical\_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool lexicographical\_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`

- `template<typename _I1, typename _I2 >`  
`bool lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare >`  
`bool lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)`
- `template<typename _Tp >`  
`complex< _Tp > log (const complex< _Tp > &)`
- `constexpr float log (float __x)`
- `constexpr long double log (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type log (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log,`  
`_ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`complex< _Tp > log10 (const complex< _Tp > &)`
- `constexpr float log10 (float __x)`
- `constexpr long double log10 (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type log10 (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log10, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log10,`  
`_ValArray, _Tp >, _Tp > log10 (const valarray< _Tp > &__v)`
- `template<typename _Filter, typename _Tp >`  
`_Filter lower\_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter lower\_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `error\_code make\_error\_code (future_errc __errc) noexcept`
- `error\_code make\_error\_code (errc __e) noexcept`
- `error\_condition make\_error\_condition (future_errc __errc) noexcept`
- `error\_condition make\_error\_condition (errc __e) noexcept`
- `template<typename _Ex >`  
`exception_ptr make\_exception\_ptr (_Ex __ex) noexcept`

- `template<typename _RAIter >`  
`void make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > make_move_iterator (_Iterator __i)`
- `template<class _T1, class _T2 >`  
`constexpr pair< typename`  
`__decay_and_strip< _T1 >`  
`::__type, typename`  
`__decay_and_strip< _T2 >`  
`::__type > make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _Tp > make_shared (_Args &&...__args)`
- `template<typename... _Elements>`  
`constexpr tuple< typename`  
`__decay_and_strip< _Elements >`  
`::__type...> make_tuple (_Elements &&...__args)`
- `template<typename _Tp >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::* __pm) noexcept`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg))`
- `void * memchr (void * __s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2`  
`__last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2`  
`__last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter min\_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min\_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`pair< _Filter, _Filter > minmax\_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`pair< _Filter, _Filter > minmax\_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator,`  
`_ForwardIterator > minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator,`  
`_ForwardIterator > minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1,`  
`_InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1,`  
`_InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Binary-`  
`Predicate __binary_pred)`
- `float modf (float __x, float *__iptr)`
- `long double modf (long double __x, long double *__iptr)`

- `template<typename _Tp >`  
`constexpr`  
`std::remove_reference<_Tp >`  
`::type && move (_Tp &&__t) noexcept`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp`  
`&, _Tp * > move (_Deque_iterator<_Tp, _Tp &, _Tp * > __first, _Deque_iterator<_Tp, _Tp &, _Tp * > __last,`  
`_Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI >`  
`_OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp`  
`&, _Tp * > move (_Deque_iterator<_Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<_Tp, const _Tp`  
`&, const _Tp * > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp`  
`&, _Tp * > move_backward (_Deque_iterator<_Tp, _Tp &, _Tp * > __first, _Deque_iterator<_Tp, _Tp &, _Tp`  
`* > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp`  
`&, _Tp * > move_backward (_Deque_iterator<_Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<_Tp,`  
`const _Tp &, const _Tp * > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`constexpr conditional`  
`<__move_if_noexcept_cond<_Tp >`  
`::value, const _Tp &, _Tp && >`  
`::type move_if_noexcept (_Tp &__x) noexcept`
- `template<typename _ForwardIterator >`  
`_ForwardIterator next (_ForwardIterator __x, typename iterator_traits<_ForwardIterator >::difference_type __`  
`n=1)`
- `template<typename _BIter >`  
`bool next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `ios_base & noboolalpha (ios_base &__base)`
- `bool none () const noexcept`
- `template<typename _Iter, typename _Predicate >`  
`bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp norm (const complex<_Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote<_Tp >::__type norm (_Tp __x)`
- `ios_base & noshowbase (ios_base &__base)`
- `ios_base & noshowpoint (ios_base &__base)`
- `ios_base & noshowpos (ios_base &__base)`

- `ios_base & noskipws (ios_base &__base)`
- `template<typename _Predicate >`  
`unary_negate< _Predicate > not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`binary_negate< _Predicate > not2 (const _Predicate &__pred)`
- `ios_base & nounitbuf (ios_base &__base)`
- `ios_base & nouppercase (ios_base &__base)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base &__base)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`  
`bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _T1, typename _T2 >`  
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`  
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`
- `bool operator!= (thread::id __x, thread::id __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`bool operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _StateT >`  
`bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<class _T1, class _T2 >`  
`constexpr bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, std::size_t _Nm >`  
`bool operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp >`  
`bool operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `template<typename _Val >`  
`bool operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `bool operator!= (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool operator!= (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `bool operator!= (const error_condition &__lhs, const error_code &__rhs) noexcept`

- `bool operator!= (const error\_condition &__lhs, const error\_condition &__rhs) noexcept`
- `template<typename _Iterator >  
bool operator!= (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _Val >  
bool operator!= (const Rb\_tree\_iterator< _Val > &__x, const Rb\_tree\_const\_iterator< _Val > &__y)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator!= (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >  
bool operator!= (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _IteratorL, typename _IteratorR >  
bool operator!= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Tp >  
bool operator!= (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>  
bool operator!= (const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos  
< __not_equal_to, _Expr, _Expr,  
_Dom1, _Dom2 >, typename __fun  
< __not_equal_to, typename  
_Dom1::value_type >  
::result_type > operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,  
typename _Dom2::value_type > &__w)`
- `template<class _Dom >  
_Expr< _BinClos  
< __not_equal_to, _Expr,  
_Constant, _Dom, typename  
_Dom::value_type >, typename  
__fun< __not_equal_to,  
typename _Dom::value_type >  
::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-  
::value_type &__t)`
- `template<class _Dom >  
_Expr< _BinClos  
< __not_equal_to, _Constant,  
_Expr, typename  
_Dom::value_type, _Dom >  
, typename __fun  
< __not_equal_to, typename  
_Dom::value_type >  
::result_type > operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-  
::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos  
< __not_equal_to, _Expr,  
_ValArray, _Dom, typename  
_Dom::value_type >, typename  
__fun< __not_equal_to,  
typename _Dom::value_type >  
::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-  
name _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos`  
`< __not_equal_to, _ValArray,`  
`_Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __not_equal_to, typename`  
`_Dom::value_type >`  
`::result_type > operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool operator!= (const scoped\_allocator\_adaptor< _OutA1, _InA...> &__a, const scoped\_allocator\_adaptor<`  
`_OutA2, _InA...> &__b) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator!= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UInt-`  
`Type __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>`  
`bool operator!= (const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b,`  
`__t, __c, __l, __f> &__lhs, const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`  
`__s, __b, __t, __c, __l, __f> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator!= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool operator!= (const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r> &__lhs, const std::subtract-`  
`with\_carry\_engine< _UIntType, __w, __s, __r> &__rhs)`
- `template<typename _Bilter >`  
`bool operator!= (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key,`  
`_Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub\_match< _Bi-`  
`iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const sub\_match< _Bi_iter > &__lhs, const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch-`  
`alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator!= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`

- `template<typename _Iterator >`  
`bool operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos`  
`< __not_equal_to, _ValArray,`  
`_ValArray, _Tp, _Tp >`  
`, typename __fun`  
`< __not_equal_to, _Tp >`  
`::result_type > operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos`  
`< __not_equal_to, _ValArray,`  
`_Constant, _Tp, _Tp >`  
`, typename __fun`  
`< __not_equal_to, _Tp >`  
`::result_type > operator!= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos`  
`< __not_equal_to, _Constant,`  
`_ValArray, _Tp, _Tp >`  
`, typename __fun`  
`< __not_equal_to, _Tp >`  
`::result_type > operator!= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _Bi_iter, class _Alloc >`  
`bool operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (const function< _Res(_Args...) > &__f, nullptr_t)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (nullptr_t, const function< _Res(_Args...) > &__f)`
- `template<typename _RealType >`  
`bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`

- `template<typename _RealType >`  
`bool operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __modulus,`  
`typename _Dom::value_type >`  
`::result_type > operator% (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun< __modulus,`  
`typename _Dom1::value_type >`  
`::result_type > operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __modulus,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __modulus,`  
`typename _Dom::value_type >`  
`::result_type > operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __modulus,`  
`typename _Dom::value_type >`  
`::result_type > operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus,`  
`_Expr, ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __modulus,`  
`typename _Dom::value_type >`  
`::result_type > operator% (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus,`  
`_Constant, ValArray, _Tp, _Tp >`  
`, typename __fun< __modulus,`  
`_Tp >::result_type > operator% (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus,`  
`ValArray, ValArray, _Tp, _Tp >`  
`, typename __fun< __modulus,`  
`_Tp >::result_type > operator% (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus,`  
`ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __modulus,`  
`_Tp >::result_type > operator% (const valarray< _Tp > &__v, const _Tp &__t)`
- `constexpr _ios_Fmtflags operator& (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr memory\_order operator& (memory\_order __m, __memory_order_modifier __mod)`
- `constexpr _ios_Openmode operator& (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr launch operator& (launch __x, launch __y)`
- `constexpr _ios_ostate operator& (_ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun`  
`< __bitwise_and, typename`  
`_Dom::value_type >`  
`::result_type > operator& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`

- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __bitwise_and,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun`  
`< __bitwise_and, typename`  
`_Dom1::value_type >`  
`::result_type > operator& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __bitwise_and, typename`  
`_Dom::value_type >`  
`::result_type > operator& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __bitwise_and, typename`  
`_Dom::value_type >`  
`::result_type > operator& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun`  
`< __bitwise_and, typename`  
`_Dom::value_type >`  
`::result_type > operator& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun`  
`< __bitwise_and, _Tp >`  
`::result_type > operator& (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun`  
`< __bitwise_and, _Tp >`  
`::result_type > operator& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun`  
`< __bitwise_and, _Tp >`  
`::result_type > operator& (const valarray< _Tp > &__v, const _Tp &__t)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_and,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun`  
`< __logical_and, typename`  
`_Dom1::value_type >`  
`::result_type > operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __logical_and, typename`  
`_Dom::value_type >`  
`::result_type > operator&& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __logical_and, typename`  
`_Dom::value_type >`  
`::result_type > operator&& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun`  
`< __logical_and, typename`  
`_Dom::value_type >`  
`::result_type > operator&& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun`  
`< __logical_and, typename`  
`_Dom::value_type >`  
`::result_type > operator&& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun`  
`< __logical_and, _Tp >`  
`::result_type > operator&& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`

```

 _Expr< _BinClos< __logical_and,
 _Constant, _ValArray, _Tp, _Tp >
 , typename __fun
 < __logical_and, _Tp >
 ::result_type > operator&& (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
 _Expr< _BinClos< __logical_and,
 _ValArray, _ValArray, _Tp, _Tp >
 , typename __fun
 < __logical_and, _Tp >
 ::result_type > operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• const _los_Fmtflags & operator&= (_los_Fmtflags &__a, _los_Fmtflags __b)
• const _los_Openmode & operator&= (_los_Openmode &__a, _los_Openmode __b)
• launch & operator&= (launch &__x, launch __y)
• const _los_losestate & operator&= (_los_losestate &__a, _los_losestate __b)
• template<class _Dom >
 _Expr< _BinClos< __multiplies,
 _Expr, _ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun< __multiplies,
 typename _Dom::value_type >
 ::result_type > operator* (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename
 _Dom::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __multiplies,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __multiplies,
 typename _Dom::value_type >
 ::result_type > operator* (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
 ::value_type > &__v)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __multiplies,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __multiplies,
 typename _Dom1::value_type >
 ::result_type > operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,
 typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __multiplies,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __multiplies,
 typename _Dom::value_type >
 ::result_type > operator* (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename
 _Dom::value_type > &__e)
• template<class _Dom >
 _Expr< _BinClos< __multiplies,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun< __multiplies,
 typename _Dom::value_type >
 ::result_type > operator* (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-
 ::value_type &__t)

```

- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __multiplies,`  
`_Tp >::result_type > operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __multiplies,`  
`_Tp >::result_type > operator* (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __multiplies,`  
`_Tp >::result_type > operator* (const _Tp &__t, const valarray< _Tp > &__v)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `template<typename _Iterator >`  
`reverse\_iterator< _Iterator > operator+ (typename reverse\_iterator< _Iterator >::difference_type __n, const`  
`reverse\_iterator< _Iterator > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`\_Deque\_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__x)`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __plus, typename`  
`_Dom::value_type >`  
`::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __plus, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename __fun< __plus,`  
`typename _Dom1::value_type >`  
`::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __plus,`  
`typename _Dom::value_type >`  
`::result_type > operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr,`  
`_ValArray, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __plus, typename`  
`_Dom::value_type >`  
`::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __plus,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __plus,`  
`typename _Dom::value_type >`  
`::result_type > operator+ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<typename _Tp >`  
`complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n, const move-`  
`iterator< _Iterator > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __plus, _Tp >`  
`::result_type > operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __plus, _Tp >`  
`::result_type > operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __plus, _Tp >`  
`::result_type > operator+ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _-`  
`Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc`  
`> &&__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc >`  
`&&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >`  
`::difference_type operator- (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref,`  
`_Ptr >::difference_type operator- (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator<`  
`_Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`_Deque_iterator< _Tp, _RefL,`  
`_PtrL >::difference_type operator- (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator<`  
`_Tp, _RefR, _PtrR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) ->`  
`decltype(__y.base()-__x.base())`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __minus,`  
`typename _Dom::value_type >`  
`::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename`  
`_Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __minus,`  
`typename _Dom::value_type >`  
`::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __minus,`  
`typename _Dom::value_type >`  
`::result_type > operator- (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value-`

```

 _type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __minus,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __minus,
 typename _Dom::value_type >
 ::result_type > operator- (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename
 _Dom::value_type > &__e)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __minus,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __minus,
 typename _Dom1::value_type >
 ::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,
 typename _Dom2::value_type > &__w)
• template<typename _Tp >
 complex< _Tp > operator- (const complex< _Tp > &__x)
• template<typename _IteratorL , typename _IteratorR >
 auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(-
 __x.base()-__y.base())
• template<typename _Iterator >
 auto operator- (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) -> decltype(-
 __x.base()-__y.base())
• template<typename _Tp >
 _Expr< _BinClos< __minus,
 _ValArray, _ValArray, _Tp, _Tp >
 , typename __fun< __minus, _Tp >
 ::result_type > operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
 _Expr< _BinClos< __minus,
 _ValArray, _Constant, _Tp, _Tp >
 , typename __fun< __minus, _Tp >
 ::result_type > operator- (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
 _Expr< _BinClos< __minus,
 _Constant, _ValArray, _Tp, _Tp >
 , typename __fun< __minus, _Tp >
 ::result_type > operator- (const _Tp &__t, const valarray< _Tp > &__v)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __divides,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __divides,
 typename _Dom1::value_type >
 ::result_type > operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,
 typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __divides,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __divides,
 typename _Dom::value_type >
 ::result_type > operator/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value-
 _type > &__v)

```

- `template<class _Dom >`  
`_Expr< _BinClos< __divides,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __divides,`  
`typename _Dom::value_type >`  
`::result_type > operator/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __divides,`  
`typename _Dom::value_type >`  
`::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __divides,`  
`typename _Dom::value_type >`  
`::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename`  
`_Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __divides,`  
`_Tp >::result_type > operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __divides,`  
`_Tp >::result_type > operator/ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __divides,`  
`_Tp >::result_type > operator/ (const _Tp &__t, const valarray< _Tp > &__v)`
- `bool operator< (const error_code &__lhs, const error_code &__rhs) noexcept`
- `template<class _T1, class _T2 >`  
`constexpr bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr >`  
`&__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR`  
`> &__y)`

- `template<typename _Tp, typename _Seq >`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`bool operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<class _Dom >`  
`_Expr< _BinClos< __less,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __less,`  
`typename _Dom::value_type >`  
`::result_type > operator< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename __fun< __less,`  
`typename _Dom1::value_type >`  
`::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __less, typename`  
`_Dom::value_type >`  
`::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __less,`  
`typename _Dom::value_type >`  
`::result_type > operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr,`  
`_ValArray, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __less, typename`  
`_Dom::value_type >`  
`::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`

- `template<typename _Tp, typename _Dp >`  
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _Bilter >`  
`bool operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Bi_iter >`  
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __less, _Tp >`  
`::result_type > operator< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __less, _Tp >`  
`::result_type > operator< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __less,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __less, _Tp >`  
`::result_type > operator< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`  
`bool operator< (const sub\_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`*__rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub\_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (const sub\_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const basic\_string< _CharT, _Traits, _`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic\_ostream< _Ch, _Tr > & operator<< (std::basic\_ostream< _Ch, _Tr > &__os, const __shared_ptr<`  
`_Tp, _Lp > &__p)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT,`  
`_Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const linear\_congruential\_engine<`  
`_UIntType, __a, __c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setfill< _CharT`  
`> __f)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, const error\_code`  
`&__e)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setprecision`  
`__f)`

- `template<class _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, __Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, __Put_money< _MoneyT > __f)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > operator<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > operator<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_left,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun< __shift_left,`  
`typename _Dom1::value_type >`  
`::result_type > operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _-`  
`Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > operator<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > operator<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UInt-`  
`Type __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _U-`  
`IntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _`  
`Tp > &__x)`

- `template<typename _CharT, typename _Traits, typename _Tp >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > && __os, const _Tp & __x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const subtract_with_carry_engine<`  
`_UIntType, __w, __s, __r > & __x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const discard_block_engine< _-`  
`RandomNumberEngine, __p, __r > & __x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const shuffle_order_engine< _-`  
`RandomNumberEngine, __k > & __x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __shift_left,`  
`_Tp >::result_type > operator<< (const valarray< _Tp > & __v, const _Tp & __t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __shift_left,`  
`_Tp >::result_type > operator<< (const _Tp & __t, const valarray< _Tp > & __v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __shift_left,`  
`_Tp >::result_type > operator<< (const valarray< _Tp > & __v, const valarray< _Tp > & __w)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::independent_bits_engine<`  
`_RandomNumberEngine, __w, _UIntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const negative_binomial_distribution<`  
`_IntType > & __x)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`  
`basic_ostream< _Ch_type,`  
`_Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > & __os, const sub_match< _Bi_iter >`  
`& __m)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const poisson_distribution< _IntType`  
`> & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const binomial_distribution< _IntType`  
`> & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::uniform_int_distribution< _IntType`  
`> & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType`  
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _Real-`  
`Type > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _Real-`  
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _-`  
`RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType`  
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _Real-`  
`Type > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx-`  
`::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType`  
`> &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_-`  
`string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType`  
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_-`  
`distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _Real-`  
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution<`  
`_RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`

- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _Int-`  
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution<`  
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _Real-`  
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT,`  
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution<`  
`_RealType > &__x)`
- `bool operator<= (thread::id __x, thread::id __y) noexcept`
- `template<class _T1, class _T2 >`  
`constexpr bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr >`  
`&__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _-`  
`PtrR > &__y)`
- `template<typename _Iterator >`  
`bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __less_equal,`  
`typename _Dom::value_type >`  
`::result_type > operator<= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<class _Dom >`

```

 _Expr< _BinClos< __less_equal,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __less_equal,
 typename _Dom::value_type >
 ::result_type > operator<= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-
 name _Dom::value_type > &__e)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __less_equal,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __less_equal,
 typename _Dom1::value_type >
 ::result_type > operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _-
 Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __less_equal,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __less_equal,
 typename _Dom::value_type >
 ::result_type > operator<= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
 ::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __less_equal,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun< __less_equal,
 typename _Dom::value_type >
 ::result_type > operator<= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-
 ::value_type &__t)
• template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
 bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
• template<typename _Tp, typename _Dp >
 bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
• template<typename _Tp, typename _Dp >
 bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
• template<typename _Key, typename _Compare, typename _Alloc >
 bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >
 &__y)
• template<typename _Key, typename _Compare, typename _Alloc >
 bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
• template<typename... _TElements, typename... _UElements>
 constexpr bool operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
• template<typename _Bilter >
 bool operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)
• template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
 bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _-
 Compare, _Alloc > &__y)
• template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >
 bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key,
 _Val, _KeyOfValue, _Compare, _Alloc > &__y)
• template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
 bool operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi-
 _iter > &__rhs)

```

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool operator<= (const sub_match< _Bi_iter > &__lhs, const sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`  
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __less_equal, _Tp >::result_type > operator<= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`  
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`  
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _T1, typename _T2 >`  
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`  
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _CharT, typename _Traits >`  
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<class _T1, class _T2 >`  
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _StateT >`  
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp >`  
`bool operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs) noexcept`
- `template<typename _Val >`  
`bool operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Iterator >`  
`bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Val >`  
`bool operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`

- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __equal_to,`  
`typename _Dom::value_type >`  
`::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to,`  
`_Expr, ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __equal_to,`  
`typename _Dom::value_type >`  
`::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __equal_to,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun< __equal_to,`  
`typename _Dom1::value_type >`  
`::result_type > operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __equal_to,`  
`typename _Dom::value_type >`  
`::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __equal_to,`  
`typename _Dom::value_type >`  
`::result_type > operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _OutA1 , typename _OutA2 , typename... _InA>`  
`bool operator== (const scoped\_allocator\_adaptor< _OutA1, _InA...> &__a, const scoped\_allocator\_adaptor<`  
`_OutA2, _InA...> &__b) noexcept`
- `template<typename _Tp , typename _Dp , typename _Up , typename _Ep >`  
`bool operator== (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp , typename _Dp >`  
`bool operator== (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp , typename _Dp >`  
`bool operator== (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`

- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator== (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _Bilter >`  
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator== (const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`  
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Bi_iter >`  
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter >`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Alloc >`  
`bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _RealType >`  
`bool operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator== (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `bool operator> (thread::id __x, thread::id __y) noexcept`
- `template<class _T1, class _T2 >`  
`constexpr bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator> (const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator> (const \_Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const \_Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`bool operator> (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator> (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator> (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator> (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom > , typename __fun< __greater, typename _Dom::value_type > ::result_type > operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type, _Dom > , typename __fun< __greater, typename _Dom::value_type > ::result_type > operator> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type > , typename __fun< __greater, typename _Dom::value_type > ::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 > , typename __fun< __greater, typename _Dom1::value_type > ::result_type > operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`

```

 _Expr< _BinClos< __greater,
 _Expr, _ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun< __greater,
 typename _Dom::value_type >
 ::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-
 name _Dom::value_type > &__v)
• template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
 bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
• template<typename _Tp, typename _Dp >
 bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
• template<typename _Tp, typename _Dp >
 bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
• template<typename _Key, typename _Compare, typename _Alloc >
 bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >
 &__y)
• template<typename _Key, typename _Compare, typename _Alloc >
 bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
• template<typename... _TElements, typename... _UElements>
 constexpr bool operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
• template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
 bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,
 _Alloc > &__y)
• template<typename _Bilter >
 bool operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)
• template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >
 bool operator> (const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb_tree< _Key,
 _Val, _KeyOfValue, _Compare, _Alloc > &__y)
• template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
 bool operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_
 iter > &__rhs)
• template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
 bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc >
 &__y)
• template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >
 bool operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_
 alloc > &__rhs)
• template<typename _IteratorL, typename _IteratorR >
 bool operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
• template<typename _Iterator >
 bool operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
• template<typename _Bi_iter >
 bool operator> (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter >
 &__rhs)
• template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>
 bool operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
• template<typename _Tp, _Lock_policy _Lp>
 bool operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
• template<typename _Tp, _Lock_policy _Lp>
 bool operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept

```

- `template<typename _Tp >`  
`_Expr< _BinClos< __greater,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, typename __fun< __greater,`  
`_Tp >::result_type > operator> (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, typename __fun< __greater,`  
`_Tp >::result_type > operator> (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, typename __fun< __greater,`  
`_Tp >::result_type > operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Bi_iter >`  
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`*__rhs)`
- `template<typename _Bi_iter >`  
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `bool operator>= (thread::id __x, thread::id __y) noexcept`
- `template<class _T1, class _T2 >`  
`constexpr bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Seq >`  
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr >`  
`&__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _`  
`PtrR > &__y)`

- `template<typename _Iterator >`  
`bool operator>= (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator>= (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _Expr,`  
`_ValArray, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __greater_equal,`  
`typename _Dom::value_type >`  
`::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __greater_equal,`  
`typename _Dom::value_type >`  
`::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`  
`::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _Constant,`  
`_Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __greater_equal, typename`  
`_Dom::value_type >`  
`::result_type > operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos`  
`< __greater_equal, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename __fun`  
`< __greater_equal, typename`  
`_Dom1::value_type >`  
`::result_type > operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _-`  
`Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _ValArray,`  
`_Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __greater_equal, typename`  
`_Dom::value_type >`  
`::result_type > operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`

```

 name _Dom::value_type > &__e)
• template<typename _Tp >
 bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
• template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
 bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
• template<typename _Tp, typename _Dp >
 bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
• template<typename _Tp, typename _Dp >
 bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
• template<typename _Key, typename _Compare, typename _Alloc >
 bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >
 &__y)
• template<typename _Key, typename _Compare, typename _Alloc >
 bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
• template<typename... _TElements, typename... _UElements>
 constexpr bool operator>= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
• template<typename _Bilter >
 bool operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)
• template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
 bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _-
 Compare, _Alloc > &__y)
• template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >
 bool operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key,
 _Val, _KeyOfValue, _Compare, _Alloc > &__y)
• template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
 bool operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi-
 _iter > &__rhs)
• template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
 bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc
 > &__y)
• template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >
 bool operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch-
 alloc > &__rhs)
• template<typename _IteratorL, typename _IteratorR >
 bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
• template<typename _Iterator >
 bool operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
• template<typename _Bi_iter >
 bool operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter >
 &__rhs)
• template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>
 bool operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
• template<typename _Tp, _Lock_policy _Lp>
 bool operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
• template<typename _Tp, _Lock_policy _Lp>
 bool operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
• template<typename _Tp >
 _Expr< _BinClos
 < __greater_equal, _ValArray,
 _ValArray, _Tp, _Tp >
 , typename __fun
 < __greater_equal, _Tp >
 ::result_type > operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

```

- `template<typename _Tp >`  
`_Expr< _BinClos`  
`< __greater_equal, _ValArray,`  
`_Constant, _Tp, _Tp >`  
`, typename __fun`  
`< __greater_equal, _Tp >`  
`::result_type > operator>= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos`  
`< __greater_equal, _Constant,`  
`_ValArray, _Tp, _Tp >`  
`, typename __fun`  
`< __greater_equal, _Tp >`  
`::result_type > operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`  
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`*__rhs)`
- `template<typename _Bi_iter >`  
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT >`  
`__f)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT,`  
`_Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType,`  
`__a, __c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)`

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_right,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun`  
`< __shift_right, typename`  
`_Dom1::value_type >`  
`::result_type > operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun`  
`< __shift_right, typename`  
`_Dom::value_type >`  
`::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __shift_right, typename`  
`_Dom::value_type >`  
`::result_type > operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun`  
`< __shift_right, typename`  
`_Dom::value_type >`  
`::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __shift_right, typename`  
`_Dom::value_type >`  
`::result_type > operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp >`  
`&__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UInt-`  
`Type __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`

- ```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType,
__w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
```
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType,
__w, __s, __r > &__x)
```
 - template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _Random-
NumberEngine, __p, __r > &__x)
```
 - template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _Random-
NumberEngine, __k > &__x)
```
 - template<typename _CharT, typename _Traits, typename _Tp >

```
basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &&__is, _Tp &__x)
```
 - template<typename _Tp >

```
_Expr< _BinClos< __shift_right,
_valArray, _ValArray, _Tp, _Tp >
, typename __fun
< __shift_right, _Tp >
::result_type > operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
```
 - template<typename _Tp >

```
_Expr< _BinClos< __shift_right,
_constant, _ValArray, _Tp, _Tp >
, typename __fun
< __shift_right, _Tp >
::result_type > operator>> (const _Tp &__t, const valarray< _Tp > &__v)
```
 - template<typename _Tp >

```
_Expr< _BinClos< __shift_right,
_valArray, _Constant, _Tp, _Tp >
, typename __fun
< __shift_right, _Tp >
::result_type > operator>> (const valarray< _Tp > &__v, const _Tp &__t)
```
 - template<typename _IntType, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution< _Int-
Type > &__x)
```
 - template<typename _IntType, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)
```
 - template<typename _IntType, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
```
 - template<typename _IntType, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
```
 - template<typename _RealType, typename _CharT, typename _Traits >

```
std::basic_istream< _CharT,
_traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType >
&)
```

- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [normal_distribution](#)< _RealType > &_
 __x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [lognormal_distribution](#)< _RealType >
 &__x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [chi_squared_distribution](#)< _RealType >
 &__x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [fisher_f_distribution](#)< _RealType > &_
 __x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [student_t_distribution](#)< _RealType >
 &__x)
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [__gnu_cxx::__versa-](#)
[string](#)< _CharT, _Traits, _Alloc, _Base > &__str)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [gamma_distribution](#)< _RealType > &_
 __x)
- `template<typename _CharT, typename _Traits, typename _Alloc >`
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [basic_string](#)< _
 CharT, _Traits, _Alloc > &__str)
- `template<>`
[basic_istream](#)< char > & **operator**>> ([basic_istream](#)< char > &__is, [basic_string](#)< char > &__str)
- `template<typename _IntType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [discrete_distribution](#)< _IntType > &__x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [std::cauchy_distribution](#)< _RealType >
 &__x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [piecewise_constant_distribution](#)< _Real-
 Type > &__x)
- `template<typename _RealType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [piecewise_linear_distribution](#)< _Real-
 Type > &__x)
- `template<typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [std::bernoulli_distribution](#) &__x)
- `template<typename _IntType, typename _CharT, typename _Traits >`
[std::basic_istream](#)< _CharT,
 _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [std::geometric_distribution](#)< _IntType >
 &__x)

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _Real-`
`Type > &__x)`
- `constexpr _ios_Fmtflags operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode operator^ (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr launch operator^ (launch __x, launch __y)`
- `constexpr _ios_ostate operator^ (_ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor,`
`_Expr, _ValArray, _Dom,`
`typename _Dom::value_type >`
`, typename __fun`
`< __bitwise_xor, typename`
`_Dom::value_type >`
`::result_type > operator^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`
`name _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor,`
`_Expr, _Constant, _Dom,`
`typename _Dom::value_type >`
`, typename __fun`
`< __bitwise_xor, typename`
`_Dom::value_type >`
`::result_type > operator^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`
`::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor,`
`_ValArray, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun`
`< __bitwise_xor, typename`
`_Dom::value_type >`
`::result_type > operator^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor,`
`_Constant, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun`
`< __bitwise_xor, typename`
`_Dom::value_type >`
`::result_type > operator^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`
`::value_type > &__v)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor,`
`_Expr, _Expr, _Dom1, _Dom2 >`
`, typename __fun`
`< __bitwise_xor, typename`
`_Dom1::value_type >`
`::result_type > operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_xor, _Tp >`
`::result_type > operator^ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_xor, _Tp >`
`::result_type > operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __bitwise_xor, _Tp >`
`::result_type > operator^ (const _Tp &__t, const valarray< _Tp > &__v)`
- `const _los_Fmtflags & operator^= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator^= (_los_Openmode &__a, _los_Openmode __b)`
- `launch & operator^= (launch &__x, launch __y)`
- `const _los_losestate & operator^= (_los_losestate &__a, _los_losestate __b)`
- `bitset< _Nb > & operator^= (const bitset< _Nb > &__rhs) noexcept`
- `constexpr memory_order operator| (memory_order __m, __memory_order_modifier __mod)`
- `constexpr _los_Fmtflags operator| (_los_Fmtflags __a, _los_Fmtflags __b)`
- `constexpr _los_Openmode operator| (_los_Openmode __a, _los_Openmode __b)`
- `constexpr launch operator| (launch __x, launch __y)`
- `constexpr _los_losestate operator| (_los_losestate __a, _los_losestate __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or,`
`_ValArray, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun< __bitwise_or,`
`typename _Dom::value_type >`
`::result_type > operator| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or,`
`_Constant, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun< __bitwise_or,`
`typename _Dom::value_type >`
`::result_type > operator| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value-`
`_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or,`
`_Expr, _Constant, _Dom,`
`typename _Dom::value_type >`
`, typename __fun< __bitwise_or,`
`typename _Dom::value_type >`
`::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-`
`::value_type &__t)`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __bitwise_or,`
`_Expr, _Expr, _Dom1, _Dom2 >`
`, typename __fun< __bitwise_or,`
`typename _Dom1::value_type >`
`::result_type > operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or,`
`_Expr, _ValArray, _Dom,`
`typename _Dom::value_type >`
`, typename __fun< __bitwise_or,`
`typename _Dom::value_type >`
`::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename`
`_Dom::value_type > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > operator| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > operator| (const _Tp &__t, const valarray< _Tp > &__v)`
- `const _ios_Fmtflags & operator|= (_ios_Fmtflags &__a, _ios_Fmtflags __b)`
- `const _ios_Openmode & operator|= (_ios_Openmode &__a, _ios_Openmode __b)`
- `launch & operator|= (launch &__x, launch __y)`
- `const _ios_istate & operator|= (_ios_istate &__a, _ios_istate __b)`
- `bitset< _Nb > & operator|= (const bitset< _Nb > &__rhs) noexcept`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __logical_or,`
`_Expr, _Expr, _Dom1, _Dom2 >`
`, typename __fun< __logical_or,`
`typename _Dom1::value_type >`
`::result_type > operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`

```

    _Expr< _BinClos< __logical_or,
    _Expr, _Constant, _Dom,
    typename _Dom::value_type >
    , typename __fun< __logical_or,
    typename _Dom::value_type >
    ::result_type > operator|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom-
    ::value_type &__t)
• template<class _Dom >
    _Expr< _BinClos< __logical_or,
    _ValArray, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __logical_or,
    typename _Dom::value_type >
    ::result_type > operator|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename
    _Dom::value_type > &__e)
• template<class _Dom >
    _Expr< _BinClos< __logical_or,
    _Expr, _ValArray, _Dom,
    typename _Dom::value_type >
    , typename __fun< __logical_or,
    typename _Dom::value_type >
    ::result_type > operator|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-
    name _Dom::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __logical_or,
    _Constant, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __logical_or,
    typename _Dom::value_type >
    ::result_type > operator|| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
    ::value_type > &__v)
• template<typename _Tp >
    _Expr< _BinClos< __logical_or,
    _ValArray, _ValArray, _Tp, _Tp >
    , typename __fun< __logical_or,
    _Tp >::result_type > operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
    _Expr< _BinClos< __logical_or,
    _Constant, _ValArray, _Tp, _Tp >
    , typename __fun< __logical_or,
    _Tp >::result_type > operator|| (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
    _Expr< _BinClos< __logical_or,
    _ValArray, _Constant, _Tp, _Tp >
    , typename __fun< __logical_or,
    _Tp >::result_type > operator|| (const valarray< _Tp > &__v, const _Tp &__t)
• constexpr _ios_Fmtflags operator~ (_ios_Fmtflags __a)
• constexpr _ios_Openmode operator~ (_ios_Openmode __a)
• constexpr launch operator~ (launch __x)
• constexpr _ios_istate operator~ (_ios_istate __a)
• bitset< _Nb > operator~ () const noexcept
• template<typename _RAIter >
    void partial_sort (_RAIter, _RAIter, _RAIter)

```

- `template<typename _RAIter, typename _Compare >`
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _RAIter >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _BIter, typename _Predicate >`
`_BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`void pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`

- constexpr float **pow** (float __x, float __y)
- constexpr long double **pow** (long double __x, long double __y)
- template<typename _Tp, typename _Up >
constexpr
__gnu_cxx::__promote_2< _Tp,
_Up >::__type **pow** (_Tp __x, _Up __y)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< _Pow, _Expr,
_Expr, _Dom1, _Dom2 >
, typename _Dom1::value_type > **pow** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const
_Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom >
_Expr< _BinClos< _Pow,
_Constant, _Expr, typename
_Dom::value_type, _Dom >
, typename _Dom::value_type > **pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename
_Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _BinClos< _Pow,
_ValArray, _Constant, _Tp, _Tp >
, _Tp > **pow** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
_Expr< _BinClos< _Pow,
_Constant, _ValArray, _Tp, _Tp >
, _Tp > **pow** (const _Tp &__t, const valarray< _Tp > &__v)
- template<class _Dom >
_Expr< _BinClos< _Pow,
_ValArray, _Expr, typename
_Dom::value_type, _Dom >
, typename _Dom::value_type > **pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom,
typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _BinClos< _Pow,
_ValArray, _ValArray, _Tp, _Tp >
, _Tp > **pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom >
_Expr< _BinClos< _Pow, _Expr,
_Constant, _Dom, typename
_Dom::value_type >, typename
_Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom-
::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< _Pow, _Expr,
_ValArray, _Dom, typename
_Dom::value_type >, typename
_Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename
_Dom::value_type > &__v)
- template<typename _Tp, typename _Up >
std::complex< typename
__gnu_cxx::__promote_2< _Tp,
_Up >::__type > **pow** (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp, typename _Up >
std::complex< typename
__gnu_cxx::__promote_2< _Tp,

- `_Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename`
`__gnu_cxx::__promote_2< _Tp,`
`_Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _BidirectionalIterator >`
`_BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::`
`difference_type __n=1)`
- `template<typename _Biter >`
`bool prev_permutation (_Biter, _Biter)`
- `template<typename _Biter, typename _Compare >`
`bool prev_permutation (_Biter, _Biter, _Compare)`
- `template<typename _BidirectionalIterator >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type proj (_Tp __x)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function`
`< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function`
`< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__x)(_Arg1, _Arg2))`
- `template<typename _RandomAccessIterator >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`void push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _MoneyT >`
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RandomAccessIterator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumber-`
`Generator &&__rand)`
- `template<typename _Tp >`
`constexpr _Tp real (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Filter, typename _Tp >`
`_Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`

- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`
`_Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`
- `bitset< _Nb > & reset () noexcept`
- `bitset< _Nb > & reset (size_t __position)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void rethrow_if_nested (const _Ex & __ex)`
- `void rethrow_if_nested (const nested_exception & __ex)`
- `template<typename _Tp >`
`void return_temporary_buffer (_Tp * __p)`
- `template<typename _BIter >`
`void reverse (_BIter, _BIter)`
- `template<typename _BidirectionalIterator >`
`void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter, typename _OIter >`
`_OIter reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `ios_base & right (ios_base & __base)`
- `template<typename _Filter >`
`void rotate (_Filter, _Filter, _Filter)`

- `template<typename _ForwardIterator >`
`void rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _Filter, typename _OIter >`
`_OIter rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _-`
`OutputIterator __result)`
- `ios_base & scientific (ios_base & __base)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _-`
`ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _-`
`ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`
`_Filter search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val,`
`_BinaryPredicate __binary_pred)`
- `bitset< _Nb > & set () noexcept`
- `bitset< _Nb > & set (size_t __position, bool __val=true)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`
`Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`
`Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input-`
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `new_handler set_new_handler (new_handler) throw ()`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `terminate_handler set_terminate (terminate_handler) noexcept`
- `unexpected_handler set_unexpected (unexpected_handler) noexcept`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT >`
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision setprecision (int __n)`
- `_Setw setw (int __n)`
- `ios_base & showbase (ios_base & __base)`
- `ios_base & showpoint (ios_base & __base)`
- `ios_base & showpos (ios_base & __base)`
- `template<typename _RAIter, typename _UGenerator >`
`void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`
- `template<typename _Tp >`
`complex< _Tp > sin (const complex< _Tp > &)`
- `constexpr float sin (float __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > sin (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp > & __v)`
- `constexpr long double sin (long double __x)`
- `template<typename _Tp >`
`constexpr`
`__gnu_cxx::__enable_if`
`< __is_integer< _Tp >::__value,`
`double >::__type sin (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > sinh (const complex< _Tp > &)`

- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh,`
`_ValArray, _Tp >, _Tp > sinh (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr,`
`_Dom >, typename`
`_Dom::value_type > sinh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float sinh (float __x)`
- `constexpr long double sinh (long double __x)`
- `template<typename _Tp >`
`constexpr`
`__gnu_cxx::__enable_if`
`< __is_integer< _Tp >::__value,`
`double >::__type sinh (_Tp __x)`
- `constexpr size_t size () const noexcept`
- `ios_base & skipws (ios_base &__base)`
- `template<typename _RAIter >`
`void sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`void sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`
`complex< _Tp > sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt,`
`_ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr,`
`_Dom >, typename`
`_Dom::value_type > sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float sqrt (float __x)`
- `constexpr long double sqrt (long double __x)`
- `template<typename _Tp >`
`constexpr`
`__gnu_cxx::__enable_if`
`< __is_integer< _Tp >::__value,`
`double >::__type sqrt (_Tp __x)`
- `template<typename _Blter, typename _Predicate >`
`_Blter stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

- `template<typename _RAIter >`
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr<_Tp> static_pointer_cast (const shared_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr<_Tp, _Lp> static_pointer_cast (const __shared_ptr<_Tp1, _Lp> &__r) noexcept`
- `char * strchr (char *__s, int __n)`
- `char * strpbrk (char *__s1, const char *__s2)`
- `char * strrchr (char *__s, int __n)`
- `char * strstr (char *__s1, const char *__s2)`
- `void swap (_Bit_reference __x, _Bit_reference __y) noexcept`
- `void swap (_Bit_reference __x, bool &__y) noexcept`
- `void swap (bool &__x, _Bit_reference __y) noexcept`
- `void swap (thread &__x, thread &__y) noexcept`
- `template<class _T1, class _T2 >`
`void swap (pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, std::size_t _Nm>`
`void swap (array<_Tp, _Nm> &__one, array<_Tp, _Nm> &__two) noexcept(noexcept(__one.swap(__two)))`
- `template<typename _Tp, typename _Seq >`
`void swap (stack<_Tp, _Seq> &__x, stack<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq >`
`void swap (queue<_Tp, _Seq> &__x, queue<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp >`
`void swap (shared_ptr<_Tp> &__a, shared_ptr<_Tp> &__b) noexcept`
- `template<typename _Tp, typename _Dp >`
`void swap (unique_ptr<_Tp, _Dp> &__x, unique_ptr<_Tp, _Dp> &__y) noexcept`
- `template<typename _Tp >`
`void swap (weak_ptr<_Tp> &__a, weak_ptr<_Tp> &__b) noexcept`
- `template<typename _Tp >`
`void swap (_Tp &__a, _Tp &__b) noexcept(__and<is_nothrow_move_constructible<_Tp>`
- `template<typename _Tp, typename _Sequence, typename _Compare >`
`void swap (priority_queue<_Tp, _Sequence, _Compare> &__x, priority_queue<_Tp, _Sequence, _Compare> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, size_t _Nm>`
`void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a`
- `template<typename _Mutex >`
`void swap (unique_lock<_Mutex> &__x, unique_lock<_Mutex> &__y) noexcept`
- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex<_Ch_type, _Rx_traits> &__lhs, basic_regex<_Ch_type, _Rx_traits> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (multiset<_Key, _Compare, _Alloc> &__x, multiset<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (set<_Key, _Compare, _Alloc> &__x, set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (multimap<_Key, _Tp, _Compare, _Alloc> &__x, multimap<_Key, _Tp, _Compare, _Alloc> &__y)`

- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename... _Elements>`
`void swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Alloc >`
`void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`
- `template<typename _Res, typename... _Args>`
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `const error_category & system_category () noexcept`
- `template<typename _Tp >`
`complex< _Tp > tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float tan (float __x)`

- constexpr long double **tan** (long double __x)
- template<typename _Tp >
constexpr
__gnu_cxx::__enable_if
< __is_integer< _Tp >::__value,
double >::__type **tan** (_Tp __x)
- template<typename _Tp >
complex< _Tp > **tanh** (const **complex**< _Tp > &)
- template<typename _Tp >
_Expr< _UnClos< _Tanh,
_ValArray, _Tp >, _Tp > **tanh** (const **valarray**< _Tp > &__v)
- template<class _Dom >
_Expr< _UnClos< _Tanh, _Expr,
_Dom >, typename
_Dom::value_type > **tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **tanh** (float __x)
- constexpr long double **tanh** (long double __x)
- template<typename _Tp >
constexpr
__gnu_cxx::__enable_if
< __is_integer< _Tp >::__value,
double >::__type **tanh** (_Tp __x)
- void **terminate** () noexcept __attribute__((__noreturn__))
- bool **test** (size_t __position) const
- template<typename _Ex >
void **throw_with_nested** (_Ex __ex)
- template<typename... _Elements>
tuple< _Elements &...> **tie** (_Elements &... __args) noexcept
- template<class _CharT, class _Traits, class _Alloc >
std::basic_string< _CharT,
_Traits, _Alloc > **to_string** () const
- template<class _CharT, class _Traits, class _Alloc >
std::basic_string< _CharT,
_Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits >
std::basic_string< _CharT,
_Traits, **std::allocator**
< _CharT > > **to_string** () const
- template<class _CharT, class _Traits >
std::basic_string< _CharT,
_Traits, **std::allocator**
< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT >
std::basic_string< _CharT,
std::char_traits< _CharT >
, **std::allocator**< _CharT > > **to_string** () const
- template<class _CharT >
std::basic_string< _CharT,
std::char_traits< _CharT >
, **std::allocator**< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- **std::basic_string**< char,
std::char_traits< char >
, **std::allocator**< char > > **to_string** (char __zero, char __one= '1') const

- unsigned long long **to_ullong** () const
- unsigned long **to_ulong** () const
- template<typename _CharT >
_CharT **tolower** (_CharT __c, const **locale** &__loc)
- template<typename _CharT >
_CharT **toupper** (_CharT __c, const **locale** &__loc)
- template<typename _Iter, typename _OIter, typename _UnaryOperation >
_OIter **transform** (_Iter, _Iter, _OIter, _UnaryOperation)
- template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >
_OIter **transform** (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >
_OutputIterator **transform** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **transform** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _Lock1, typename _Lock2, typename... _Lock3>
int **try_lock** (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &... __l3)
- template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>
constexpr auto **tuple_cat** (_Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::type
- bool **uncaught_exception** () noexcept __attribute__((__pure__))
- void **unexpected** () __attribute__((__noreturn__))
- template<typename _InputIterator, typename _ForwardIterator >
_ForwardIterator **uninitialized_copy** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator >
_ForwardIterator **uninitialized_copy_n** (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator, typename _Tp >
void **uninitialized_fill** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp >
void **uninitialized_fill_n** (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _Filter >
_Filter **unique** (_Filter, _Filter)
- template<typename _Filter, typename _BinaryPredicate >
_Filter **unique** (_Filter, _Filter, _BinaryPredicate)
- template<typename _ForwardIterator >
_ForwardIterator **unique** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate >
_ForwardIterator **unique** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _Iter, typename _OIter >
_OIter **unique_copy** (_Iter, _Iter, _OIter)
- template<typename _Iter, typename _OIter, typename _BinaryPredicate >
_OIter **unique_copy** (_Iter, _Iter, _OIter, _BinaryPredicate)
- template<typename _InputIterator, typename _OutputIterator >
_OutputIterator **unique_copy** (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >
_OutputIterator **unique_copy** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- **ios_base** & **unitbuf** (**ios_base** &__base)
- template<typename _Filter, typename _Tp >
_Filter **upper_bound** (_Filter, _Filter, const _Tp &)
- template<typename _Filter, typename _Tp, typename _Compare >
_Filter **upper_bound** (_Filter, _Filter, const _Tp &, _Compare)

- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `ios_base & uppercase (ios_base &__base)`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &__loc)`
- `wchar_t * wcschr (wchar_t * __p, wchar_t __c)`
- `wchar_t * wcsbrk (wchar_t * __s1, const wchar_t * __s2)`
- `wchar_t * wcsrchr (wchar_t * __p, wchar_t __c)`
- `wchar_t * wcsstr (wchar_t * __s1, const wchar_t * __s2)`
- `wchar_t * wmemchr (wchar_t * __p, wchar_t __c, size_t __n)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > &__is)`
- `bitset< _Nb > & operator<<= (size_t __position) noexcept`
- `bitset< _Nb > & operator>>= (size_t __position) noexcept`
- `bitset< _Nb > & _Unchecked_set (size_t __pos) noexcept`
- `bitset< _Nb > & _Unchecked_set (size_t __pos, int __val) noexcept`
- `bitset< _Nb > & _Unchecked_reset (size_t __pos) noexcept`
- `bitset< _Nb > & _Unchecked_flip (size_t __pos) noexcept`
- `constexpr bool _Unchecked_test (size_t __pos) const noexcept`
- reference `operator[] (size_t __position)`
- `bool operator== (const bitset< _Nb > &__rhs) const noexcept`
- `bool operator!= (const bitset< _Nb > &__rhs) const noexcept`
- `bitset< _Nb > operator<< (size_t __position) const noexcept`
- `bitset< _Nb > operator>> (size_t __position) const noexcept`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x, const _Tp &__y)`

- `template<typename _Tp >`
`complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool operator!= (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (_Tp &__t) noexcept`
- `template<typename _Tp >`
`reference_wrapper< const _Tp > cref (const _Tp &__t) noexcept`
- `template<typename _Tp >`
`void ref (const _Tp &&)=delete`
- `template<typename _Tp >`
`void cref (const _Tp &&)=delete`
- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t) noexcept`
- `template<typename _Tp >`
`reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t) noexcept`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
- `template<>`
`basic_istream< char > & operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`

- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_`
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_`
`default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_`
`regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_`
`default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_`
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex<`
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_`
`type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_`
`traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_`
`constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _`
`Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type`
`__flags=regex_constants::match_default)`

Variables

- `__a`
- `void * __b`
- `static ios_base::Init __ioint`
- `function< void()> __once_functor`
- `void`
`is_nothrow_move_assignable`
`< _Tp >::value _Tp __tmp`
- `constexpr adopt_lock_t adopt_lock`
- `constexpr allocator_arg_t allocator_arg`
- `constexpr defer_lock_t defer_lock`
- `const _Swallow_assign ignore`
- `error_code make_error_code (errc) noexcept`

- [error_condition](#) **make_error_condition** (errc) noexcept
- const nothrow_t **nothrow**
- decltype(nullptr) typedef **nullptr_t**
- constexpr [piecewise_construct_t](#) **piecewise_construct**
- return * **this**
- constexpr [try_to_lock_t](#) **try_to_lock**

Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the HOWTO linked to above.

- [istream](#) **cin**
- [ostream](#) **cout**
- [ostream](#) **cerr**
- [ostream](#) **clog**
- [wistream](#) **wcin**
- [wostream](#) **wcout**
- [wostream](#) **wcerr**
- [wostream](#) **wclog**

3.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

3.11.2 Typedef Documentation

3.11.2.1 `template<bool _Cache> using std::__umap_traits = typedef __detail::_Hashtable_traits<_Cache, false, true>`

Base types for unordered_map.

Definition at line 39 of file unordered_map.h.

3.11.2.2 `template<bool _Cache> using std::__ummap_traits = typedef __detail::_Hashtable_traits<_Cache, false, false>`

Base types for unordered_multimap.

Definition at line 56 of file unordered_map.h.

3.11.2.3 `template<bool _Cache> using std::__umset_traits = typedef __detail::_Hashtable_traits<_Cache, true, false>`

Base types for unordered_multiset.

Definition at line 54 of file unordered_set.h.

3.11.2.4 `template<bool _Cache> using std::__uset_traits = typedef __detail::_Hashtable_traits<_Cache, true, true>`

Base types for unordered_set.

Definition at line 39 of file unordered_set.h.

3.11.2.5 `typedef void(* std::new_handler)()`

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 73 of file new.

3.11.2.6 typedef long long std::streamoff

Type used by fpos, char_traits<char>, and char_traits<wchar_t>.

In clauses 21.1.3.1 and 27.4.1 streamoff is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, streamoff was typedef long.

Definition at line 94 of file postypes.h.

3.11.2.7 typedef fpos<mbstate_t> std::streampos

File position for char streams.

Definition at line 228 of file postypes.h.

3.11.2.8 typedef ptrdiff_t std::streamsize

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file postypes.h.

3.11.2.9 typedef fpos<mbstate_t> std::u16streampos

File position for char16_t streams.

Definition at line 234 of file postypes.h.

3.11.2.10 typedef fpos<mbstate_t> std::u32streampos

File position for char32_t streams.

Definition at line 236 of file postypes.h.

3.11.2.11 typedef fpos<mbstate_t> std::wstreampos

File position for wchar_t streams.

Definition at line 230 of file postypes.h.

3.11.3 Enumeration Type Documentation

3.11.3.1 anonymous enum

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html> This controls some aspect of the sort routines.

Definition at line 2225 of file stl_algo.h.

3.11.3.2 enum std::float_denorm_style

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the std::numeric_limits class.

Enumerator:

denorm_indeterminate Indeterminate at compile time whether denormalized values are allowed.

denorm_absent The type does not allow denormalized values.

denorm_present The type allows denormalized values.

Definition at line 171 of file limits.

3.11.3.3 enum std::float_round_style

Describes the rounding style for floating-point types.

This is used in the std::numeric_limits class.

Enumerator:

round_toward_zero Intermediate.
round_to_nearest To zero.
round_toward_infinity To the nearest representable value.
round_toward_neg_infinity To infinity.

Definition at line 156 of file limits.

3.11.4 Function Documentation

3.11.4.1 `template<typename _RandomAccessIterator > void std::__final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2230 of file stl_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `sort()`.

3.11.4.2 `template<typename _RandomAccessIterator, typename _Compare > void std::__final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2245 of file stl_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

3.11.4.3 `template<typename _InputIterator, typename _Tp > _InputIterator std::__find (_InputIterator __first, _InputIterator __last, const _Tp & __val, input_iterator_tag) [inline]`

This is an overload used by `find()` for the Input Iterator case.

Definition at line 130 of file stl_algo.h.

Referenced by `find()`.

3.11.4.4 `template<typename _RandomAccessIterator, typename _Tp > _RandomAccessIterator std::__find (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __val, random_access_iterator_tag)`

This is an overload used by `find()` for the RAI case.

Definition at line 152 of file stl_algo.h.

3.11.4.5 `template<typename _InputIterator, typename _Predicate > _InputIterator std::__find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag) [inline]`

This is an overload used by `find_if()` for the Input Iterator case.

Definition at line 141 of file `stl_algo.h`.

Referenced by `find_if()`.

```
3.11.4.6  template<typename _RandomAccessIterator, typename _Predicate > _RandomAccessIterator std::_find_if (
        _RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag )
```

This is an overload used by `find_if()` for the RAI case.

Definition at line 200 of file `stl_algo.h`.

```
3.11.4.7  template<typename _InputIterator, typename _Predicate > _InputIterator std::_find_if_not ( _InputIterator __first,
        _InputIterator __last, _Predicate __pred, input_iterator_tag ) [inline]
```

This is an overload used by `find_if_not()` for the Input Iterator case.

Definition at line 248 of file `stl_algo.h`.

Referenced by `__find_if_not()`, `find_if_not()`, and `stable_partition()`.

```
3.11.4.8  template<typename _RandomAccessIterator, typename _Predicate > _RandomAccessIterator std::_find_if_not (
        _RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag )
```

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 259 of file `stl_algo.h`.

```
3.11.4.9  template<typename _InputIterator, typename _Predicate > _InputIterator std::_find_if_not ( _InputIterator __first,
        _InputIterator __last, _Predicate __pred ) [inline]
```

Provided for `stable_partition` to use.

Definition at line 307 of file `stl_algo.h`.

References `__find_if_not()`, and `__iterator_category()`.

```
3.11.4.10 template<typename _InputIterator, typename _Predicate, typename _Distance > _InputIterator std::_find_if_not_n (
        _InputIterator __first, _Distance & __len, _Predicate __pred )
```

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Definition at line 319 of file `stl_algo.h`.

Referenced by `__inplace_stable_partition()`, and `__stable_partition_adaptive()`.

```
3.11.4.11 template<typename _EuclideanRingElement > _EuclideanRingElement std::_gcd ( _EuclideanRingElement __m,
        _EuclideanRingElement __n )
```

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1518 of file `stl_algo.h`.

```
3.11.4.12 template<typename _RandomAccessIterator > void std::_heap_select ( _RandomAccessIterator __first,
        _RandomAccessIterator __middle, _RandomAccessIterator __last )
```

This is a helper function for the sort routines.

Definition at line 1953 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

3.11.4.13 `template<typename _RandomAccessIterator, typename _Compare> void std::__heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routines.

Definition at line 1966 of file `stl_algo.h`.

References `make_heap()`.

3.11.4.14 `template<typename _ForwardIterator, typename _Predicate, typename _Distance> _ForwardIterator std::__inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`

This is a helper function... Requires `__len != 0` and `!__pred(*__first)`, same as `__stable_partition_adaptive`.

Definition at line 1809 of file `stl_algo.h`.

References `__find_if_not_n()`, `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

3.11.4.15 `template<typename _RandomAccessIterator> void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

This is a helper function for the stable sorting routines.

Definition at line 3509 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

3.11.4.16 `template<typename _RandomAccessIterator, typename _Compare> void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the stable sorting routines.

Definition at line 3528 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insertion_sort()`, and `__merge_without_buffer()`.

3.11.4.17 `template<typename _RandomAccessIterator> void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2153 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

3.11.4.18 `template<typename _RandomAccessIterator, typename _Compare> void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2176 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

3.11.4.19 `template<typename _RandomAccessIterator, typename _Size> void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`

This is a helper function for the sort routine.

Definition at line 2325 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

Referenced by `__introsort_loop()`, and `sort()`.

3.11.4.20 `template<typename _RandomAccessIterator, typename _Size, typename _Compare> void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2347 of file `stl_algo.h`.

References `__introsort_loop()`, and `__unguarded_partition_pivot()`.

3.11.4.21 `template<typename _Functor, typename... _Args> enable_if<(is_member_pointer<_Functor>::value && !is_function<_Functor>::value && !is_function<typename remove_pointer<_Functor>::type>::value), typename result_of<_Functor(_Args &&...)>::type>::type std::__invoke (_Functor & __f, _Args &&... __args) [inline]`

Invoke a function object, which may be either a member pointer or a function object. The first parameter will tell which.

Definition at line 232 of file `functional`.

3.11.4.22 `constexpr int std::__lg (int __n) [inline]`

This is a helper function for the sort routines and for `random.tcc`.

Definition at line 980 of file `stl_algbase.h`.

Referenced by `nth_element()`, `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator()()`, `std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed()`, and `sort()`.

3.11.4.23 `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer> void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2957 of file `stl_algo.h`.

References `__move_merge_adaptive()`, `__move_merge_adaptive_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

3.11.4.24 `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare> void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 3013 of file `stl_algo.h`.

References `__merge_adaptive()`, `__move_merge_adaptive()`, `__move_merge_adaptive_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

3.11.4.25 `template<typename _BidirectionalIterator, typename _Distance> void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2)`

This is a helper function for the merge routines.

Definition at line 3070 of file `stl_algo.h`.

References `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

Referenced by `__inplace_stable_sort()`, `__merge_without_buffer()`, and `inplace_merge()`.

3.11.4.26 `template<typename _BidirectionalIterator, typename _Distance, typename _Compare> void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 3114 of file `stl_algo.h`.

References `__merge_without_buffer()`, `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

3.11.4.27 `template<typename _Iterator> void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__a`.

Definition at line 78 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

3.11.4.28 `template<typename _Iterator, typename _Compare> void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__a`.

Definition at line 102 of file `stl_algo.h`.

References `iter_swap()`.

3.11.4.29 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

This is a helper function for the `__merge_sort_loop` routines.

Definition at line 3272 of file `stl_algo.h`.

3.11.4.30 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> _OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

This is a helper function for the `__merge_sort_loop` routines.

Definition at line 3299 of file `stl_algo.h`.

3.11.4.31 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2778 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

```
3.11.4.32 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > void
std::__move_merge_adaptive ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result, _Compare __comp )
```

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2804 of file `stl_algo.h`.

```
3.11.4.33 template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3
> void std::__move_merge_adaptive_backward ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1,
_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result )
```

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2830 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

```
3.11.4.34 template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 , typename
_Compare > void std::__move_merge_adaptive_backward ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1,
_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp )
```

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2872 of file `stl_algo.h`.

```
3.11.4.35 template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::__partition ( _ForwardIterator __first,
_FowardIterator __last, _Predicate __pred, forward_iterator_tag )
```

This is a helper function...

Definition at line 1752 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

```
3.11.4.36 template<typename _BidirectionalIterator , typename _Predicate > _BidirectionalIterator std::__partition (
_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag )
```

This is a helper function...

Definition at line 1777 of file `stl_algo.h`.

References `iter_swap()`.

```
3.11.4.37 template<typename _BidirectionalIterator > void std::__reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last,
bidirectional_iterator_tag )
```

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1418 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__rotate()`, and `reverse()`.

3.11.4.38 `template<typename _RandomAccessIterator > void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1438 of file `stl_algo.h`.

References `iter_swap()`.

3.11.4.39 `template<typename _ForwardIterator > void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1532 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`, and `rotate()`.

3.11.4.40 `template<typename _BidirectionalIterator > void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1568 of file `stl_algo.h`.

References `__reverse()`, and `iter_swap()`.

3.11.4.41 `template<typename _RandomAccessIterator > void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1598 of file `stl_algo.h`.

References `iter_swap()`, and `swap_ranges()`.

3.11.4.42 `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance > _BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2915 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

3.11.4.43 `template<typename _ForwardIterator, typename _Integer, typename _Tp > _ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 347 of file `stl_algo.h`.

Referenced by `search_n()`.

```
3.11.4.44 template<typename _RandomAccessIter, typename _Integer, typename _Tp > _RandomAccessIter
std::__search_n( _RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp & __val,
std::random_access_iterator_tag )
```

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&) overloaded for random access iterators.

Definition at line 379 of file stl_algo.h.

```
3.11.4.45 template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate > _ForwardIterator
std::__search_n( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::forward_iterator_tag )
```

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate) overloaded for forward iterators.

Definition at line 433 of file stl_algo.h.

```
3.11.4.46 template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate >
_RandomAccessIter std::__search_n( _RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp
& __val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag )
```

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate) overloaded for random access iterators.

Definition at line 472 of file stl_algo.h.

```
3.11.4.47 template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance > _ForwardIterator
std::__stable_partition_adaptive( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len,
_PPointer __buffer, _Distance __buffer_size )
```

This is a helper function... Requires __first != __last and !__pred(*__first) and __len == distance(__first, __last).

!__pred(*__first) allows us to guarantee that we don't move-assign an element onto itself.

Definition at line 1841 of file stl_algo.h.

References __find_if_not_n(), advance(), distance(), and rotate().

Referenced by stable_partition().

```
3.11.4.48 template<typename _RandomAccessIterator > void std::__unguarded_insertion_sort( _RandomAccessIterator __first,
_RandomAccessIterator __last ) [inline]
```

This is a helper function for the sort routine.

Definition at line 2198 of file stl_algo.h.

References __unguarded_linear_insert().

Referenced by __final_insertion_sort().

```
3.11.4.49 template<typename _RandomAccessIterator, typename _Compare > void std::__unguarded_insertion_sort(
_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp ) [inline]
```

This is a helper function for the sort routine.

Definition at line 2211 of file stl_algo.h.

References __unguarded_linear_insert().

3.11.4.50 `template<typename _RandomAccessIterator > void std::__unguarded_linear_insert (_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2116 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

3.11.4.51 `template<typename _RandomAccessIterator , typename _Compare > void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2134 of file `stl_algo.h`.

3.11.4.52 `template<typename _RandomAccessIterator , typename _Tp > _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __pivot)`

This is a helper function...

Definition at line 2261 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

3.11.4.53 `template<typename _RandomAccessIterator , typename _Tp , typename _Compare > _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp & __pivot, _Compare __comp)`

This is a helper function...

Definition at line 2281 of file `stl_algo.h`.

References `iter_swap()`.

3.11.4.54 `template<typename _RandomAccessIterator > _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

This is a helper function...

Definition at line 2302 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

3.11.4.55 `template<typename _RandomAccessIterator , typename _Compare > _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

This is a helper function...

Definition at line 2314 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

3.11.4.56 `template<typename _ForwardIterator , typename _OutputIterator > _OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag , output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output iterator as result.

Definition at line 1270 of file `stl_algo.h`.

Referenced by `unique_copy()`.

```
3.11.4.57 template<typename _InputIterator, typename _OutputIterator > _OutputIterator std::__unique_copy ( _InputIterator __first,
    _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output iterator as result.

Definition at line 1293 of file `stl_algo.h`.

```
3.11.4.58 template<typename _InputIterator, typename _ForwardIterator > _ForwardIterator std::__unique_copy ( _InputIterator
    __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward iterator as result.

Definition at line 1316 of file `stl_algo.h`.

```
3.11.4.59 template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate > _OutputIterator
    std::__unique_copy ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate
    __binary_pred, forward_iterator_tag, output_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1337 of file `stl_algo.h`.

```
3.11.4.60 template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate > _OutputIterator
    std::__unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred,
    input_iterator_tag, output_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1366 of file `stl_algo.h`.

```
3.11.4.61 template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate > _ForwardIterator
    std::__unique_copy ( _InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred,
    input_iterator_tag, forward_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1395 of file `stl_algo.h`.

```
3.11.4.62 template<typename _T1, typename... _Args> void std::__Construct ( _T1 * __p, _Args &&... __args ) [inline]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 74 of file `stl_construct.h`.

```
3.11.4.63 template<typename _Tp> void std::__Destroy ( _Tp * __pointer ) [inline]
```

Destroy the object pointed to by a pointer type.

Definition at line 92 of file `stl_construct.h`.

Referenced by `std::deque< _Tp, _Alloc >::M_fill_initialize()`, `std::deque< _Tp, _Alloc >::M_range_initialize()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::reserve()`, and `std::vector< sub_match< _Bi_iter >`,

allocator< sub_match< _Bi_iter > > >::~~vector().

3.11.4.64 `template<typename _ForwardIterator > void std::Destroy (_ForwardIterator __first, _ForwardIterator __last)`
`[inline]`

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 122 of file stl_construct.h.

3.11.4.65 `template<typename _InputIterator, typename _Tp > _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)` `[inline]`

Accumulate values in a range.

Accumulates the values in the range [first,last) using operator+(). The initial value is *init*. The values are processed in order.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

Returns

The final sum.

Definition at line 120 of file stl_numeric.h.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

3.11.4.66 `template<typename _InputIterator, typename _Tp, typename _BinaryOperation > _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)` `[inline]`

Accumulate values in a range with operation.

Accumulates the values in the range [first,last) using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	Function object to accumulate with.

Returns

The final sum.

Definition at line 146 of file stl_numeric.h.

3.11.4.67 `template<typename _Tp > std::complex< _Tp > std::acos (const std::complex< _Tp > & __z)` `[inline]`

`acos(__z)` [8.1.2].

Definition at line 1610 of file complex.

3.11.4.68 `template<typename _Tp> std::complex<_Tp> std::acosh (const std::complex<_Tp> & __z) [inline]`

`acosh(__z)` [8.1.5].

Definition at line 1726 of file `complex`.

3.11.4.69 `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[first,last)` using operator-() and writes the result to `__result`.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sums.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 317 of file `stl_numeric.h`.

3.11.4.70 `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[__first,__last)` using the function object `__binary_op` and writes the result to `__result`.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 360 of file `stl_numeric.h`.

3.11.4.71 `template<typename _InputIterator, typename _Distance> void std::advance (_InputIterator & __i, _Distance __n) [inline]`

A generalization of pointer arithmetic.

Parameters

<code>__i</code>	An input iterator.
<code>__n</code>	The <i>delta</i> by which to change <code>__i</code> .

Returns

Nothing.

This increments `i` by `n`. For bidirectional and random access iterators, `__n` may be negative, in which case `__i` is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 173 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::M_range_initialize()`, `equal_range()`, `__gnu_pbds::detail::pat_tribase::Node_citer<Node, Leaf, Head, Inode, Cliterator, Iterator, _Alloc>::get_child()`, `__gnu_pbds::detail::pat_tribase::Node_iter<Node, Leaf, Head, Inode, Cliterator, Iterator, _Alloc>::get_child()`, `is_permutation()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

3.11.4.72 `bool std::all() const` [noexcept]

Tests whether all the bits are on.

Returns

True if all the bits are set.

Definition at line 1317 of file `bitset`.

3.11.4.73 `bool std::any() const` [noexcept]

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

Definition at line 1325 of file `bitset`.

3.11.4.74 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::arg(_Tp __x)` [inline]

Additional overloads [8.1.9].

Definition at line 1824 of file `complex`.

References `arg()`.

3.11.4.75 `template<typename _Tp> std::complex<_Tp> std::asin(const std::complex<_Tp> & __z)` [inline]

`asin(__z)` [8.1.3].

Definition at line 1646 of file `complex`.

3.11.4.76 `template<typename _Tp> std::complex<_Tp> std::asinh (const std::complex<_Tp> & __z) [inline]`

`asinh(__z)` [8.1.6].

Definition at line 1765 of file `complex`.

3.11.4.77 `template<typename _Tp> std::complex<_Tp> std::atan (const std::complex<_Tp> & __z) [inline]`

`atan(__z)` [8.1.4].

Definition at line 1690 of file `complex`.

3.11.4.78 `template<typename _Tp> std::complex<_Tp> std::atanh (const std::complex<_Tp> & __z) [inline]`

`atanh(__z)` [8.1.7].

Definition at line 1809 of file `complex`.

3.11.4.79 `template<class _Container> auto std::begin (_Container & __cont)-> decltype(__cont.begin()) [inline]`

Return an iterator pointing to the first element of the container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 48 of file `range_access.h`.

3.11.4.80 `template<class _Container> auto std::begin (const _Container & __cont)-> decltype(__cont.begin()) [inline]`

Return an iterator pointing to the first element of the const container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 58 of file `range_access.h`.

3.11.4.81 `template<class _Tp, size_t _Nm> _Tp* std::begin (_Tp(&) __arr[_Nm]) [inline]`

Return an iterator pointing to the first element of the array.

Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 87 of file `range_access.h`.

3.11.4.82 `template<class _Tp> constexpr const _Tp* std::begin (initializer_list<_Tp> __ils) [noexcept]`

Return an iterator pointing to the first element of the `initializer_list`.

Parameters

<code>__ils</code>	Initializer list.
--------------------	-------------------

Definition at line 89 of file `initializer_list`.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::deque< _Tp, _Alloc >::erase()`, `__gnu_pbds::detail::pat-trie_base::_Node_iter< Node, Leaf, Head, Inode, _Cliterator, Iterator, _Alloc >::get_child()`, `std::vector< _Tp, _Alloc >::insert()`, `std::list< _Tp, _Alloc >::merge()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::remove()`, `std::list< _Tp, _Alloc >::remove_if()`, `std::list< _Tp, _Alloc >::resize()`, `std::list< _Tp, _Alloc >::sort()`, `std::forward_list< _Tp, _Alloc >::unique()`, and `std::list< _Tp, _Alloc >::unique()`.

3.11.4.83 `ios_base& std::boolalpha (ios_base & __base) [inline]`

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 795 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::boolalpha`.

3.11.4.84 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> _shared_ptr<_Tp, _Lp> std::const_pointer_cast (const _shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]`

`const_pointer_cast`

Definition at line 1196 of file `shared_ptr_base.h`.

3.11.4.85 `size_t std::count () const [noexcept]`

Returns the number of bits which are set.

Definition at line 1277 of file `bitset`.

Referenced by `is_permutation()`.

3.11.4.86 `template<typename _Tp> reference_wrapper<const _Tp> std::cref (const _Tp & __t) [inline], [noexcept]`

Denotes a const reference should be taken to a variable.

Definition at line 481 of file `functional`.

Referenced by `cref()`.

3.11.4.87 `template<typename _Tp> void std::cref (const _Tp &&) [delete]`

Denotes a reference should be taken to a variable.

3.11.4.88 `template<typename _Tp> reference_wrapper<const _Tp> std::cref (reference_wrapper< _Tp > __t) [inline], [noexcept]`

Partial specialization.

Definition at line 499 of file `functional`.

References `cref()`.

3.11.4.89 `ios_base& std::dec (ios_base & __base) [inline]`

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 933 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::dec`, and `std::ios_base::setf()`.

Referenced by `operator>>()`.

3.11.4.90 `template<typename _InputIterator > iterator_traits<_InputIterator>::difference_type std::distance (_InputIterator __first, _InputIterator __last) [inline]`

A generalization of pointer arithmetic.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 114 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::_M_range_initialize()`, `equal_range()`, `inplace_merge()`, `is_heap_until()`, `is_permutation()`, `std::sub_match<_Bi_iter>::length()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_pbds::detail::pat_trie_base::_Node_citer<Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc>::num_children()`, `partition_point()`, `std::match_results<_FwdIterT, _Alloc>::position()`, `__gnu_cxx::random_sample_n()`, `std::list<__inp, __rebind_inp>::size()`, and `upper_bound()`.

3.11.4.91 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (const __shared_ptr<_Tp1, _Lp> & __r) [inline], [noexcept]`

`dynamic_pointer_cast`

Definition at line 1206 of file `shared_ptr_base.h`.

3.11.4.92 `template<class _Container > auto std::end (_Container & __cont)-> decltype(__cont.end()) [inline]`

Return an iterator pointing to one past the last element of the container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 68 of file `range_access.h`.

3.11.4.93 `template<class _Container > auto std::end (const _Container & __cont)-> decltype(__cont.end()) [inline]`

Return an iterator pointing to one past the last element of the const container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 78 of file `range_access.h`.

3.11.4.94 `template<class _Tp, size_t _Nm> _Tp* std::end (_Tp(&) __arr[_Nm]) [inline]`

Return an iterator pointing to one past the last element of the array.

Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 97 of file `range_access.h`.

3.11.4.95 `template<class _Tp> constexpr const _Tp* std::end (initializer_list< _Tp> __ils) [noexcept]`

Return an iterator pointing to one past the last element of the `initializer_list`.

Parameters

<code>__ils</code>	Initializer list.
--------------------	-------------------

Definition at line 99 of file `initializer_list`.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::vector< _Tp, _Alloc >::erase()`, `std::deque< _Tp, _Alloc >::erase()`, `std::vector< _Tp, _Alloc >::insert()`, `std::list< _Tp, _Alloc >::merge()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::remove()`, `std::list< _Tp, _Alloc >::remove_if()`, `std::list< _Tp, _Alloc >::resize()`, `std::forward_list< _Tp, _Alloc >::resize()`, `std::forward_list< _Tp, _Alloc >::unique()`, and `std::list< _Tp, _Alloc >::unique()`.

3.11.4.96 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>& std::endl (basic_ostream<_CharT, _Traits> & __os) [inline]`

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this subject.

Definition at line 564 of file `ostream`.

References `flush()`, `std::basic_ostream< _CharT, _Traits >::put()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

3.11.4.97 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>& std::ends (basic_ostream<_CharT, _Traits> & __os) [inline]`

Write a null character into the output sequence.

Null character is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

Definition at line 576 of file `ostream`.

References `std::basic_ostream< _CharT, _Traits >::put()`.

3.11.4.98 `template<typename _Tp> _Tp std::fabs (const std::complex< _Tp> & __z) [inline]`

`fabs(__z)` [8.1.8].

Definition at line 1818 of file `complex`.

References `abs()`.

3.11.4.99 ios_base& std::fixed (ios_base & __base) [inline]

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 958 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::fixed`, `std::ios_base::floatfield`, and `std::ios_base::setf()`.

3.11.4.100 bitset<_Nb>& std::flip () [noexcept]

Toggles every bit to its opposite value.

Definition at line 1104 of file `bitset`.

3.11.4.101 bitset<_Nb>& std::flip (size_t __position)

Toggles a given bit to its opposite value.

Parameters

<code>__position</code>	The index of the bit.
-------------------------	-----------------------

Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is bigger the size of the set.
--------------------------------	--

Definition at line 1117 of file `bitset`.

References `_Unchecked_flip()`.

3.11.4.102 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>& std::flush (basic_ostream<_CharT, _Traits> & __os) [inline]

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 586 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits>::flush()`.

Referenced by `endl()`.

3.11.4.103 template<typename _MoneyT> _Get_money<_MoneyT> std::get_money (_MoneyT & __mon, bool __intl = false) [inline]

Extended manipulator for extracting money.

Parameters

<code>__mon</code>	Either long double or a specialization of <code>basic_string</code> .
<code>__intl</code>	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts `__mon`.

Definition at line 256 of file `iomanip`.

3.11.4.104 template<typename _Tp> pair<_Tp*, ptrdiff_t> std::get_temporary_buffer (ptrdiff_t __len) [noexcept]

Allocates a temporary buffer.

Parameters

<code>__len</code>	The number of objects of type <code>Tp</code> .
--------------------	---

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A pair<> is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 85 of file `std_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.

3.11.4.105 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

Definition at line 626 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::max_size()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

3.11.4.106 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits>& std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Read a line from stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Definition at line 2795 of file basic_string.h.

References `getline()`, and `std::basic_ios<_CharT, _Traits >::widen()`.

3.11.4.109 `ios_base& std::hex (ios_base & __base) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 941 of file ios_base.h.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::hex`, and `std::ios_base::setf()`.

Referenced by `std::regex_traits<_Ch_type >::value()`.

3.11.4.110 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp > _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init) [inline]`

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using `operator+()`. The values in the ranges are processed in order.

Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.

Returns

The final inner product.

Definition at line 174 of file stl_numeric.h.

3.11.4.111 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 > _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2) [inline]`

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	Function object to accumulate with.
<code>__binary_op2</code>	Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 206 of file stl_numeric.h.

3.11.4.112 `ios_base& std::internal (ios_base & __base) [inline]`

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 908 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::internal`.

3.11.4.113 `template<typename _ForwardIterator, typename _Tp> void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

Returns

Nothing.

Definition at line 82 of file `std_numeric.h`.

3.11.4.114 `template<typename _CharT> bool std::isalnum (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

Definition at line 2584 of file `locale_facets.h`.

3.11.4.115 `template<typename _CharT> bool std::isalpha (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

Definition at line 2560 of file `locale_facets.h`.

3.11.4.116 `template<typename _CharT> bool std::iscntrl (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

Definition at line 2542 of file `locale_facets.h`.

3.11.4.117 `template<typename _CharT> bool std::isdigit (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

Definition at line 2566 of file `locale_facets.h`.

3.11.4.118 `template<typename _CharT> bool std::isgraph (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

Definition at line 2590 of file `locale_facets.h`.

3.11.4.119 `template<typename _CharT> bool std::islower (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

Definition at line 2554 of file `locale_facets.h`.

3.11.4.120 `template<typename _CharT > bool std::isprint (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

Definition at line 2536 of file `locale_facets.h`.

3.11.4.121 `template<typename _CharT > bool std::ispunct (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

Definition at line 2572 of file `locale_facets.h`.

3.11.4.122 `template<typename _CharT > bool std::isspace (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

Definition at line 2530 of file `locale_facets.h`.

3.11.4.123 `template<typename _CharT > bool std::isupper (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

Definition at line 2548 of file `locale_facets.h`.

3.11.4.124 `template<typename _CharT > bool std::isxdigit (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

Definition at line 2578 of file `locale_facets.h`.

3.11.4.125 `ios_base& std::left (ios_base & __base) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 916 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, `std::ios_base::left`, and `std::ios_base::setf()`.

Referenced by operator<<().

3.11.4.126 `ios_base& std::noboolalpha (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::boolalpha)`.

Definition at line 803 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::boolalpha`, and `std::ios_base::unsetf()`.

3.11.4.127 `bool std::none () const [noexcept]`

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

Definition at line 1333 of file `bitset`.

3.11.4.128 `ios_base& std::noshowbase (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::showbase)`.

Definition at line 819 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::showbase`, and `std::ios_base::unsetf()`.

3.11.4.129 `ios_base& std::noshowpoint (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::showpoint)`.

Definition at line 835 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::showpoint`, and `std::ios_base::unsetf()`.

3.11.4.130 `ios_base& std::noshowpos (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::showpos)`.

Definition at line 851 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::showpos`, and `std::ios_base::unsetf()`.

3.11.4.131 `ios_base& std::noskipws (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::skipws)`.

Definition at line 867 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::skipws`, and `std::ios_base::unsetf()`.

3.11.4.132 `ios_base& std::nounitbuf (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::unitbuf)`.

Definition at line 899 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::unitbuf`, and `std::ios_base::unsetf()`.

3.11.4.133 `ios_base& std::nouppercase (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::uppercase)`.

Definition at line 883 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::unsetf()`, and `std::ios_base::uppercase`.

3.11.4.134 `ios_base& std::oct (ios_base & __base) [inline]`

Calls `base.setf(ios_base::oct, ios_base::basefield)`.

Definition at line 949 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::oct`, and `std::ios_base::setf()`.

Referenced by `std::regex_traits< _Ch_type >::value()`.

3.11.4.135 `template<typename _Tp, typename _Seq> bool std::operator!= (const stack< _Tp, _Seq> & __x, const stack< _Tp, _Seq> & __y) [inline]`

Based on `operator==`.

Definition at line 267 of file `stl_stack.h`.

3.11.4.136 `template<typename _Tp> bool std::operator!= (const _Fwd_list_iterator< _Tp> & __x, const _Fwd_list_const_iterator< _Tp> & __y) [inline]`

Forward list iterator inequality comparison.

Definition at line 267 of file forward_list.h.

```
3.11.4.137 template<typename _Tp, typename _Seq > bool std::operator!= ( const queue< _Tp, _Seq > & __x, const queue< _Tp,
    _Seq > & __y ) [inline]
```

Based on operator==.

Definition at line 292 of file stl_queue.h.

```
3.11.4.138 template<typename _Key, typename _Compare, typename _Alloc > bool std::operator!= ( const multiset< _Key,
    _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns !(x == y).

Definition at line 761 of file stl_multiset.h.

```
3.11.4.139 template<typename _Key, typename _Compare, typename _Alloc > bool std::operator!= ( const set< _Key, _Compare,
    _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns !(x == y).

Definition at line 776 of file stl_set.h.

```
3.11.4.140 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator!= ( const
    multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 887 of file stl_multimap.h.

```
3.11.4.141 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator!= ( const map<
    _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 985 of file stl_map.h.

```
3.11.4.142 bool std::operator!= ( const bitset< _Nb > & __rhs ) const [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1292 of file bitset.

```
3.11.4.143 template<typename _Tp, typename _Alloc > bool std::operator!= ( const forward_list< _Tp, _Alloc > & __x, const
    forward_list< _Tp, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1367 of file forward_list.h.

```
3.11.4.144 template<typename _Tp, typename _Alloc > bool std::operator!= ( const vector< _Tp, _Alloc > & __x, const vector<
    _Tp, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1423 of file stl_vector.h.

```
3.11.4.145 template<typename _Tp, typename _Alloc > bool std::operator!= ( const list< _Tp, _Alloc > & __x, const list< _Tp,
    _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1634 of file `stl_list.h`.

```
3.11.4.146 template<typename _Tp, typename _Alloc > bool std::operator!=( const deque< _Tp, _Alloc > & __x, const deque<
    _Tp, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1983 of file `stl_deque.h`.

```
3.11.4.147 template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator!=( const basic_string< _CharT,
    _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test difference of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2532 of file `basic_string.h`.

```
3.11.4.148 template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator!=( const _CharT * __lhs, const
    basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test difference of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2544 of file `basic_string.h`.

```
3.11.4.149 template<typename _Res, typename... _Args> bool std::operator!=( const function< _Res(_Args...)> & __f, nullptr_t )
    [inline], [noexcept]
```

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`false` if the wrapper has no target, `true` otherwise

This function will not throw an exception.

Definition at line 2552 of file `functional`.

```
3.11.4.150 template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator!=( const basic_string< _CharT,
    _Traits, _Alloc > & __lhs, const _CharT * __rhs ) [inline]
```

Test difference of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2556 of file `basic_string.h`.

3.11.4.151 `template<typename _Res, typename... _Args> bool std::operator!=(nullptr_t, const function< _Res(_Args...)> & _f)`
`[inline], [noexcept]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2558 of file `functional`.

3.11.4.152 `template<size_t _Nb> bitset<_Nb> std::operator& (const bitset< _Nb > & __x, const bitset< _Nb > & __y)`
`[inline], [noexcept]`

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1416 of file `bitset`.

3.11.4.153 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string<_CharT, _Traits, _Alloc>`
`std::operator+ (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &`
`__rhs)`

Concatenate two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 2365 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`.

3.11.4.154 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >
std::operator+ (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`

Concatenate C string and string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 692 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

3.11.4.155 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >
std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`

Concatenate character and string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 708 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

3.11.4.156 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string<_CharT, _Traits, _Alloc>
std::operator+ (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]`

Concatenate string and C string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 2402 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.

```
3.11.4.157 template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>
std::operator+ ( const basic_string<_CharT, _Traits, _Alloc> & __lhs, _CharT __rhs ) [inline]
```

Concatenate string and character.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 2418 of file `basic_string.h`.

```
3.11.4.158 template<typename _Tp, typename _Seq> bool std::operator< ( const stack<_Tp, _Seq> & __x, const stack<_Tp,
_Seq> & __y ) [inline]
```

Stack ordering relation.

Parameters

<code>__x</code>	A stack.
<code>__y</code>	A stack of the same type as <code>x</code> .

Returns

True iff `x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 261 of file `stl_stack.h`.

```
3.11.4.159 template<typename _Tp, typename _Seq> bool std::operator< ( const queue<_Tp, _Seq> & __x, const queue<_Tp,
_Seq> & __y ) [inline]
```

Queue ordering relation.

Parameters

<code>__x</code>	A queue.
<code>__y</code>	A queue of the same type as <code>x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 286 of file `std_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

3.11.4.160 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator< (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]`

Multiset ordering relation.

Parameters

<code>__x</code>	A multiset.
<code>__y</code>	A multiset of the same type as <code>__x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 754 of file `std_multiset.h`.

3.11.4.161 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator< (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Set ordering relation.

Parameters

<code>__x</code>	A set.
<code>__y</code>	A set of the same type as <code>x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 769 of file `std_set.h`.

3.11.4.162 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Multimap ordering relation.

Parameters

<code>__x</code>	A multimap.
<code>__y</code>	A multimap of the same type as <code>__x</code> .

Returns

True iff `x` is lexicographically less than `y`.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 880 of file `stl_multimap.h`.

```
3.11.4.163 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator< ( const map<
    _Key, _Tp, _Compare, _Alloc> & __x, const map< _Key, _Tp, _Compare, _Alloc> & __y ) [inline]
```

Map ordering relation.

Parameters

<code>__x</code>	A map.
<code>__y</code>	A map of the same type as <code>x</code> .

Returns

True iff `x` is lexicographically less than `y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 978 of file `stl_map.h`.

```
3.11.4.164 template<typename _Tp, typename _Alloc> bool std::operator< ( const forward_list< _Tp, _Alloc> & __lx, const
    forward_list< _Tp, _Alloc> & __ly ) [inline]
```

Forward list ordering relation.

Parameters

<code>__lx</code>	A <code>forward_list</code> .
<code>__ly</code>	A <code>forward_list</code> of the same type as <code>__lx</code> .

Returns

True iff `__lx` is lexicographically less than `__ly`.

This is a total ordering relation. It is linear in the number of elements of the forward lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1359 of file `forward_list.h`.

References `lexicographical_compare()`.

3.11.4.165 `template<typename _Tp, typename _Alloc > bool std::operator< (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]`

Vector ordering relation.

Parameters

<code>__x</code>	A vector.
<code>__y</code>	A vector of the same type as <code>__x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1416 of file `stl_vector.h`.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, and `lexicographical_compare()`.

3.11.4.166 `template<typename _Tp, typename _Alloc > bool std::operator< (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]`

List ordering relation.

Parameters

<code>__x</code>	A list.
<code>__y</code>	A list of the same type as <code>__x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1627 of file `stl_list.h`.

References `lexicographical_compare()`.

3.11.4.167 `template<typename _Tp, typename _Alloc > bool std::operator< (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Deque ordering relation.

Parameters

<code>__x</code>	A deque.
<code>__y</code>	A deque of the same type as <code>__x</code> .

Returns

True iff *x* is lexicographically less than *__y*.

This is a total ordering relation. It is linear in the size of the dequeues. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1975 of file `stl_deque.h`.

References `lexicographical_compare()`.

3.11.4.168 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const basic_string< _CharT, _Traits, _Alloc> & __lhs, const basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if string precedes string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2569 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::compare()`.

3.11.4.169 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const basic_string< _CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test if string precedes C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2581 of file `basic_string.h`.

3.11.4.170 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const _CharT* __lhs, const basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string precedes string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2593 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.171 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> & __out, _CharT __c) [inline]`

Character inserters.

Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 471 of file `ostream`.

3.11.4.172 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> & __out, char __c) [inline]`

Character inserters.

Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 476 of file `ostream`.

3.11.4.173 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream<char, _Traits> & __out, char __c) [inline]`

Character inserters.

Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 482 of file `ostream`.

3.11.4.174 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits> & __out, signed char __c) [inline]`

Character inserters.

Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 488 of file `ostream`.

3.11.4.175 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits> & __out, unsigned char __c) [inline]`

Character inserters.

Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 493 of file ostream.

3.11.4.176 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> & __out, const _CharT * __s) [inline]`

String inserters.

Parameters

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

Returns

`out`

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 513 of file ostream.

References `std::ios_base::badbit`.

3.11.4.177 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> & __out, const char * __s)`

String inserters.

Parameters

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

Returns

`out`

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 321 of file ostream.tcc.

References `std::ios_base::badbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

3.11.4.178 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream<char, _Traits> & __out, const char * __s) [inline]`

String inserters.

Parameters

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

Returns

`out`

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 530 of file `ostream`.

References `std::ios_base::badbit`.

3.11.4.179 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits> & __out, const signed char * __s) [inline]`

String inserters.

Parameters

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

Returns

`out`

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 543 of file `ostream`.

3.11.4.180 `template<class _Traits> basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits> & __out, const unsigned char * __s) [inline]`

String inserters.

Parameters

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

Returns

out

Precondition`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 548 of file ostream.

3.11.4.181 `template<typename _CharT, typename _Traits, typename _Tp> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> && __os, const _Tp & __x) [inline]`

Generic inserter for rvalue stream.

Parameters

<code>__os</code>	An input stream.
<code>__x</code>	A reference to the object being inserted.

Returns

os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 602 of file ostream.

3.11.4.182 `bitset<_Nb> std::operator<< (size_t __position) const [noexcept]`

Self-explanatory.

Definition at line 1339 of file bitset.

3.11.4.183 `template<class _CharT, class _Traits, size_t _Nb> std::basic_ostream<_CharT, _Traits>& std::operator<< (std::basic_ostream<_CharT, _Traits> & __os, const bitset<_Nb> & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

Definition at line 1521 of file bitset.

References `std::__ctype_abstract_base<_CharT>::widen()`.

3.11.4.184 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> & __os, const __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Write string to a stream.

Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2521 of file `vstring.h`.

```
3.11.4.185 template<typename _CharT, typename _Traits, typename _Alloc> basic_ostream<_CharT, _Traits>&
std::operator<< ( basic_ostream<_CharT, _Traits> & __os, const basic_string<_CharT, _Traits, _Alloc> & __str )
[inline]
```

Write string to a stream.

Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2753 of file `basic_string.h`.

```
3.11.4.186 bitset<_Nb>& std::operator<=< ( size_t __position ) [noexcept]
```

Operations on bitsets.

Parameters

<code>__position</code>	The number of places to shift.
-------------------------	--------------------------------

These should be self-explanatory.

Definition at line 980 of file `bitset`.

```
3.11.4.187 template<typename _Tp, typename _Seq> bool std::operator<= ( const stack<_Tp, _Seq> & __x, const stack<_Tp,
_Seq> & __y ) [inline]
```

Based on `operator<`.

Definition at line 279 of file `stl_stack.h`.

```
3.11.4.188 template<typename _Tp, typename _Seq> bool std::operator<= ( const queue<_Tp, _Seq> & __x, const queue<
_Tp, _Seq> & __y ) [inline]
```

Based on `operator<`.

Definition at line 304 of file `stl_queue.h`.

3.11.4.189 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator<= (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(y < x)

Definition at line 775 of file `stl_multiset.h`.

3.11.4.190 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator<= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(y < x)

Definition at line 790 of file `stl_set.h`.

3.11.4.191 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 901 of file `stl_multimap.h`.

3.11.4.192 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 999 of file `stl_map.h`.

3.11.4.193 `template<typename _Tp , typename _Alloc > bool std::operator<= (const forward_list< _Tp, _Alloc > & __x, const forward_list< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1388 of file `forward_list.h`.

3.11.4.194 `template<typename _Tp , typename _Alloc > bool std::operator<= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1435 of file `stl_vector.h`.

3.11.4.195 `template<typename _Tp , typename _Alloc > bool std::operator<= (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1646 of file `stl_list.h`.

3.11.4.196 `template<typename _Tp , typename _Alloc > bool std::operator<= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1997 of file `stl_deque.h`.

3.11.4.197 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if string doesn't follow string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2643 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.198 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test if string doesn't follow C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2655 of file `basic_string.h`.

3.11.4.199 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= (const _CharT* __lhs, const basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string doesn't follow string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2667 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.200 `template<typename _StateT> bool std::operator== (const fpos< _StateT> & __lhs, const fpos< _StateT> & __rhs) [inline]`

Test if equivalent to another position.

Definition at line 216 of file `postypes.h`.

3.11.4.201 `template<typename _Tp, typename _Seq> bool std::operator== (const stack< _Tp, _Seq> & __x, const stack< _Tp, _Seq> & __y) [inline]`

Stack equality comparison.

Parameters

<code>__x</code>	A stack.
<code>__y</code>	A stack of the same type as <code>__x</code> .

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 243 of file `stl_stack.h`.

```
3.11.4.202 template<typename _Tp > bool std::operator== ( const _Fwd_list_iterator< _Tp > & __x, const
_Fwd_list_const_iterator< _Tp > & __y ) [inline]
```

Forward list iterator equality comparison.

Definition at line 258 of file `forward_list.h`.

```
3.11.4.203 template<typename _Tp, typename _Seq > bool std::operator== ( const queue< _Tp, _Seq > & __x, const queue< _Tp,
_Seq > & __y ) [inline]
```

Queue equality comparison.

Parameters

<code>__x</code>	A queue.
<code>__y</code>	A queue of the same type as <code>__x</code> .

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 268 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

```
3.11.4.204 template<typename _Key, typename _Compare, typename _Alloc > bool std::operator== ( const multiset< _Key,
_Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Multiset equality comparison.

Parameters

<code>__x</code>	A multiset.
<code>__y</code>	A multiset of the same type as <code>__x</code> .

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 737 of file `stl_multiset.h`.

```
3.11.4.205 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator==( const set< _Key, _Compare,
    _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Set equality comparison.

Parameters

<code>__x</code>	A set.
<code>__y</code>	A set of the same type as <code>x</code> .

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 752 of file `stl_set.h`.

```
3.11.4.206 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator==( const
    multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Multimap equality comparison.

Parameters

<code>__x</code>	A multimap.
<code>__y</code>	A multimap of the same type as <code>__x</code> .

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 863 of file `stl_multimap.h`.

```
3.11.4.207 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator==( const map<
    _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map equality comparison.

Parameters

<code>__x</code>	A map.
<code>__y</code>	A map of the same type as <code>x</code> .

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 961 of file `std_map.h`.

3.11.4.208 `bool std::operator==(const bitset< _Nb > & __rhs) const` `[noexcept]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1288 of file `bitset`.

3.11.4.209 `template<typename _Tp, typename _Alloc > bool std::operator==(const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly)`

Forward list equality comparison.

Parameters

<code>__lx</code>	A <code>forward_list</code>
<code>__ly</code>	A <code>forward_list</code> of the same type as <code>__lx</code> .

Returns

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 387 of file `forward_list.tcc`.

References `std::forward_list< _Tp, _Alloc >::cbegin()`, and `std::forward_list< _Tp, _Alloc >::cend()`.

3.11.4.210 `template<typename _Tp, typename _Alloc > bool std::operator==(const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y)` `[inline]`

Vector equality comparison.

Parameters

<code>__x</code>	A vector.
<code>__y</code>	A vector of the same type as <code>__x</code> .

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1399 of file `std_vector.h`.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equal()`, and `std::vector< _Tp, _Alloc >::size()`.

3.11.4.211 `template<typename _Tp, typename _Alloc > bool std::operator==(const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]`

List equality comparison.

Parameters

<code>__x</code>	A list.
<code>__y</code>	A list of the same type as <code>__x</code> .

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1598 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

3.11.4.212 `template<typename _Tp, typename _Alloc > bool std::operator==(const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Deque equality comparison.

Parameters

<code>__x</code>	A deque.
<code>__y</code>	A deque of the same type as <code>__x</code> .

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1957 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `equal()`, and `std::deque< _Tp, _Alloc >::size()`.

3.11.4.213 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator==(const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test equivalence of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2486 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.214 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==(const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test equivalence of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2507 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.215 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==(const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test equivalence of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2519 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.216 `template<typename _Res, typename... _Args> bool std::operator==(const function<_Res(_Args...)> & __f, nullptr_t) [inline], [noexcept]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 2534 of file `functional`.

3.11.4.217 `template<typename _Res, typename... _Args> bool std::operator==(nullptr_t, const function< _Res(_Args...)> & _f)`
`[inline], [noexcept]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2540 of file functional.

3.11.4.218 `template<typename _Tp, typename _Seq > bool std::operator> (const stack< _Tp, _Seq > & _x, const stack< _Tp, _Seq > & _y)` `[inline]`

Based on operator<.

Definition at line 273 of file stl_stack.h.

3.11.4.219 `template<typename _Tp, typename _Seq > bool std::operator> (const queue< _Tp, _Seq > & _x, const queue< _Tp, _Seq > & _y)` `[inline]`

Based on operator<.

Definition at line 298 of file stl_queue.h.

3.11.4.220 `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator> (const multiset< _Key, _Compare, _Alloc > & _x, const multiset< _Key, _Compare, _Alloc > & _y)` `[inline]`

Returns $y < x$.

Definition at line 768 of file stl_multiset.h.

3.11.4.221 `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator> (const set< _Key, _Compare, _Alloc > & _x, const set< _Key, _Compare, _Alloc > & _y)` `[inline]`

Returns $y < x$.

Definition at line 783 of file stl_set.h.

3.11.4.222 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > & _x, const multimap< _Key, _Tp, _Compare, _Alloc > & _y)` `[inline]`

Based on operator<.

Definition at line 894 of file stl_multimap.h.

3.11.4.223 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > & _x, const map< _Key, _Tp, _Compare, _Alloc > & _y)` `[inline]`

Based on operator<.

Definition at line 992 of file stl_map.h.

3.11.4.224 `template<typename _Tp, typename _Alloc > bool std::operator> (const forward_list< _Tp, _Alloc > & _x, const forward_list< _Tp, _Alloc > & _y)` `[inline]`

Based on operator<.

Definition at line 1374 of file forward_list.h.

3.11.4.225 `template<typename _Tp, typename _Alloc > bool std::operator> (const vector< _Tp, _Alloc > & _x, const vector< _Tp, _Alloc > & _y)` `[inline]`

Based on operator<.

Definition at line 1429 of file `stl_vector.h`.

```
3.11.4.226 template<typename _Tp, typename _Alloc > bool std::operator> ( const list< _Tp, _Alloc > & __x, const list< _Tp,
    _Alloc > & __y ) [inline]
```

Based on `operator<`.

Definition at line 1640 of file `stl_list.h`.

```
3.11.4.227 template<typename _Tp, typename _Alloc > bool std::operator> ( const deque< _Tp, _Alloc > & __x, const deque<
    _Tp, _Alloc > & __y ) [inline]
```

Based on `operator<`.

Definition at line 1990 of file `stl_deque.h`.

```
3.11.4.228 template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> ( const basic_string< _CharT,
    _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if string follows string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2606 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

```
3.11.4.229 template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> ( const basic_string< _CharT,
    _Traits, _Alloc > & __lhs, const _CharT* __rhs ) [inline]
```

Test if string follows C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2618 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

```
3.11.4.230 template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> ( const _CharT* __lhs, const
    basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string follows string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2630 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.231 `template<typename _Tp, typename _Seq> bool std::operator>= (const stack< _Tp, _Seq> & __x, const stack< _Tp, _Seq> & __y) [inline]`

Based on `operator<`.

Definition at line 285 of file `stl_stack.h`.

3.11.4.232 `template<typename _Tp, typename _Seq> bool std::operator>= (const queue< _Tp, _Seq> & __x, const queue< _Tp, _Seq> & __y) [inline]`

Based on `operator<`.

Definition at line 310 of file `stl_queue.h`.

3.11.4.233 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator>= (const multiset< _Key, _Compare, _Alloc> & __x, const multiset< _Key, _Compare, _Alloc> & __y) [inline]`

Returns `!(x < y)`

Definition at line 782 of file `stl_multiset.h`.

3.11.4.234 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator>= (const set< _Key, _Compare, _Alloc> & __x, const set< _Key, _Compare, _Alloc> & __y) [inline]`

Returns `!(x < y)`

Definition at line 797 of file `stl_set.h`.

3.11.4.235 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc> & __x, const multimap< _Key, _Tp, _Compare, _Alloc> & __y) [inline]`

Based on `operator<`.

Definition at line 908 of file `stl_multimap.h`.

3.11.4.236 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc> & __x, const map< _Key, _Tp, _Compare, _Alloc> & __y) [inline]`

Based on `operator<`.

Definition at line 1006 of file `stl_map.h`.

3.11.4.237 `template<typename _Tp, typename _Alloc> bool std::operator>= (const forward_list< _Tp, _Alloc> & __lx, const forward_list< _Tp, _Alloc> & __ly) [inline]`

Based on `operator<`.

Definition at line 1381 of file `forward_list.h`.

3.11.4.238 `template<typename _Tp, typename _Alloc> bool std::operator>= (const vector< _Tp, _Alloc> & __x, const vector< _Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1441 of file `stl_vector.h`.

3.11.4.239 `template<typename _Tp, typename _Alloc> bool std::operator>= (const list< _Tp, _Alloc> & __x, const list< _Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1652 of file `stl_list.h`.

3.11.4.240 `template<typename _Tp, typename _Alloc> bool std::operator>= (const deque< _Tp, _Alloc> & __x, const deque< _Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 2004 of file `stl_deque.h`.

3.11.4.241 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc> & __lhs, const basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if string doesn't precede string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2680 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::compare()`.

3.11.4.242 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test if string doesn't precede C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2692 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::compare()`.

3.11.4.243 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string doesn't precede string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2704 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::compare()`.

3.11.4.244 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::operator>> (basic_istream< _CharT, _Traits> & __in, _CharT & __c)`

Character extractors.

Parameters

<code>__in</code>	An input stream.
<code>__c</code>	A character reference.

Returns

`in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

Definition at line 923 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

3.11.4.245 `template<class _Traits> basic_istream<char, _Traits>& std::operator>> (basic_istream< char, _Traits> & __in, unsigned char & __c) [inline]`

Character extractors.

Parameters

<code>__in</code>	An input stream.
<code>__c</code>	A character reference.

Returns

`in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit

in the input stream.

Definition at line 727 of file istream.

3.11.4.246 `template<class _Traits> basic_istream<char, _Traits>& std::operator>> (basic_istream< char, _Traits> & __in, signed char & __c) [inline]`

Character extractors.

Parameters

<code>__in</code>	An input stream.
<code>__c</code>	A character reference.

Returns

`in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

Definition at line 732 of file istream.

3.11.4.247 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::operator>> (basic_istream< _CharT, _Traits> & __in, _CharT * __s)`

Character string extractors.

Parameters

<code>__in</code>	An input stream.
<code>__s</code>	A pointer to a character array.

Returns

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 955 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::getloc()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

3.11.4.248 `template<> basic_istream<char>& std::operator>> (basic_istream< char > & __in, char * __s)`

Character string extractors.

Parameters

<code>__in</code>	An input stream.
<code>__s</code>	A pointer to a character array.

Returns

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

3.11.4.249 `template<class _Traits > basic_istream<char, _Traits>& std::operator>> (basic_istream< char, _Traits > & __in, unsigned char * __s) [inline]`

Character string extractors.

Parameters

<code>__in</code>	An input stream.
<code>__s</code>	A pointer to a character array.

Returns`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 774 of file `istream`.

3.11.4.250 `template<class _Traits> basic_istream<char, _Traits>& std::operator>> (basic_istream< char, _Traits> & __in, signed char * __s) [inline]`

Character string extractors.

Parameters

<code>__in</code>	An input stream.
<code>__s</code>	A pointer to a character array.

Returns`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 779 of file `istream`.

3.11.4.251 `template<typename _CharT, typename _Traits, typename _Tp> basic_istream<_CharT, _Traits>& std::operator>> (basic_istream<_CharT, _Traits> && __is, _Tp & __x) [inline]`

Generic extractor for rvalue stream.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A reference to the extraction target.

Returns

`is`

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 872 of file `istream`.

3.11.4.252 `bitset<_Nb> std::operator>> (size_t __position) const [noexcept]`

Self-explanatory.

Definition at line 1343 of file `bitset`.

3.11.4.253 `template<class _CharT, class _Traits, size_t _Nb> std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream<_CharT, _Traits> & __is, bitset<_Nb> & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept `0` and `1` characters, and will only extract as many digits as the bitset will hold.

Definition at line 1453 of file `bitset`.

References `std::basic_string<_CharT, _Traits, _Alloc>::empty()`, `std::basic_string<_CharT, _Traits, _Alloc>::push_back()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_ios<_CharT, _Traits>::widen()`.

3.11.4.254 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str)`

Read stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 551 of file `vstring.tcc`.

References `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::ios_base::width()`.

3.11.4.255 `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream< _CharT, _Traits> & std::operator>> (basic_istream< _CharT, _Traits> & __is, basic_string< _CharT, _Traits, _Alloc> & __str)`

Read stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 996 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc>::append()`, `std::basic_string< _CharT, _Traits, _Alloc>::erase()`, `std::ios_base::getloc()`, `std::basic_string< _CharT, _Traits, _Alloc>::max_size()`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

3.11.4.256 `bitset<_Nb>& std::operator>>= (size_t __position) [noexcept]`

Operations on bitsets.

Parameters

<code>__position</code>	The number of places to shift.
-------------------------	--------------------------------

These should be self-explanatory.

Definition at line 993 of file `bitset`.

3.11.4.257 `constexpr bool std::operator[] (size_t __position)`

Array-indexing support.

Parameters

<code>__position</code>	Index into the bitset.
-------------------------	------------------------

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

_GLIBCXX_RESOLVE_LIB_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1145 of file bitset.

```
3.11.4.258 template<size_t _Nb> bitset<_Nb> std::operator^ ( const bitset< _Nb > & __x, const bitset< _Nb > & __y )
[inline], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1434 of file bitset.

```
3.11.4.259 template<size_t _Nb> bitset<_Nb> std::operator| ( const bitset< _Nb > & __x, const bitset< _Nb > & __y )
[inline], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1425 of file bitset.

```
3.11.4.260 bitset<_Nb> std::operator~ ( ) const [noexcept]
```

See the no-argument flip().

Definition at line 1126 of file `bitset`.

3.11.4.261 `template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

Return list of partial sums.

Accumulates the values in the range `[first,last)` using the `+` operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.

Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 237 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs_pu()`, and `__gnu_parallel::__sequential_random_shuffle()`.

3.11.4.262 `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

Return list of partial sums.

Accumulates the values in the range `[first,last)` using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 278 of file `stl_numeric.h`.

3.11.4.263 `template<typename _MoneyT> _Put_money<_MoneyT> std::put_money (const _MoneyT & __mon, bool __intl = false) [inline]`

Extended manipulator for inserting money.

Parameters

<code>__mon</code>	Either long double or a specialization of <code>basic_string</code> .
<code>__intl</code>	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts `__mon`.

Definition at line 303 of file `iomanip`.

3.11.4.264 `template<typename _Tp> reference_wrapper<_Tp> std::ref (_Tp & __t) [inline], [noexcept]`

Denotes a reference should be taken to a variable.

Definition at line 475 of file `functional`.

Referenced by `ref()`.

3.11.4.265 `template<typename _Tp> void std::ref (const _Tp &&) [delete]`

Denotes a reference should be taken to a variable.

3.11.4.266 `template<typename _Tp> reference_wrapper<_Tp> std::ref (reference_wrapper<_Tp> __t) [inline], [noexcept]`

Partial specialization.

Definition at line 493 of file `functional`.

References `ref()`.

3.11.4.267 `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __old_value, const _Tp & __new_value)`

Copy a sequence, replacing each element of one value with another value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[__first,__last)` to the output range `[__result,__result+(__last-__first))` replacing elements equal to `__old_value` with `__new_value`.

Definition at line 3907 of file `stl_algo.h`.

3.11.4.268 `bitset<_Nb>& std::reset () [noexcept]`

Sets every bit to false.

Definition at line 1079 of file `bitset`.

3.11.4.269 `bitset<_Nb>& std::reset (size_t __position)`

Sets a given bit to false.

Parameters

<code>__position</code>	The index of the bit.
-------------------------	-----------------------

Exceptions

<code>std::out_of_range</code>	If <i>pos</i> is bigger the size of the set.
--------------------------------	--

Same as writing `set(pos, false)`.

Definition at line 1093 of file `bitset`.

References `_Unchecked_reset()`.

3.11.4.270 `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask) [inline]`

Manipulator for `setf`.

Parameters

<code>__mask</code>	A format flags mask.
---------------------	----------------------

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0, __mask)`.

Definition at line 63 of file `iomanip`.

3.11.4.271 `template<typename _Tp> void std::return_temporary_buffer (_Tp * __p) [inline]`

The companion to `get_temporary_buffer()`.

Parameters

<code>__p</code>	A buffer previously allocated by <code>get_temporary_buffer</code> .
------------------	--

Returns

None.

Frees the memory pointed to by `__p`.

Definition at line 112 of file `std_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.

3.11.4.272 `ios_base& std::right (ios_base & __base) [inline]`

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 924 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, `std::ios_base::right`, and `std::ios_base::setf()`.

3.11.4.273 `ios_base& std::scientific (ios_base & __base) [inline]`

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 966 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::floatfield`, `std::ios_base::scientific`, and `std::ios_base::setf()`.

Referenced by `operator<<()`.

3.11.4.274 `bitset<_Nb>& std::set () [noexcept]`

Sets every bit to true.

Definition at line 1054 of file `bitset`.

3.11.4.275 `bitset<_Nb>& std::set (size_t __position, bool __val = true)`

Sets a given bit to a particular value.

Parameters

<code>__position</code>	The index of the bit.
<code>__val</code>	Either true or false, defaults to true.

Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is bigger the size of the set.
--------------------------------	--

Definition at line 1068 of file `bitset`.

References `_Unchecked_set()`.

3.11.4.276 `new_handler std::set_new_handler (new_handler) throw ()`

Takes a replacement handler as the argument, returns the previous handler.

3.11.4.277 `_Setbase std::setbase (int __base) [inline]`

Manipulator for `setf`.

Parameters

<code>__base</code>	A numeric base.
---------------------	-----------------

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when `base` is 8, 10, or 16, accordingly, and to 0 if `__base` is any other value.

Definition at line 124 of file `iomanip`.

3.11.4.278 `template<typename _CharT> _Setfill<_CharT> std::setfill (_CharT __c) [inline]`

Manipulator for `fill`.

Parameters

<code>__c</code>	The new fill character.
------------------	-------------------------

Sent to a stream object, this manipulator calls `fill (__c)` for that object.

Definition at line 162 of file `iomanip`.

3.11.4.279 `_Setiosflags std::setiosflags (ios_base::fmtflags __mask) [inline]`

Manipulator for `setf`.

Parameters

<code>__mask</code>	A format flags mask.
---------------------	----------------------

Sent to a stream object, this manipulator sets the format flags to `__mask`.

Definition at line 93 of file `iomanip`.

3.11.4.280 `_Setprecision` `std::setprecision (int __n)` `[inline]`

Manipulator for `precision`.

Parameters

<code>__n</code>	The new precision.
------------------	--------------------

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

Definition at line 192 of file `iomanip`.

3.11.4.281 `_Setw` `std::setw (int __n)` `[inline]`

Manipulator for `width`.

Parameters

<code>__n</code>	The new width.
------------------	----------------

Sent to a stream object, this manipulator calls `width(__n)` for that object.

Definition at line 222 of file `iomanip`.

3.11.4.282 `ios_base&` `std::showbase (ios_base & __base)` `[inline]`

Calls `base.setf(ios_base::showbase)`.

Definition at line 811 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::showbase`.

3.11.4.283 `ios_base&` `std::showpoint (ios_base & __base)` `[inline]`

Calls `base.setf(ios_base::showpoint)`.

Definition at line 827 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::showpoint`.

3.11.4.284 `ios_base&` `std::showpos (ios_base & __base)` `[inline]`

Calls `base.setf(ios_base::showpos)`.

Definition at line 843 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::showpos`.

3.11.4.285 `constexpr size_t` `std::size ()` `const` `[noexcept]`

Returns the total number of bits.

Definition at line 1282 of file `bitset`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`, `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`, `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::deque< _Tp, _Alloc >::erase()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `std::basic_string< _CharT, _Traits, _Alloc >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_of()`,

__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_of(), std::vector< _Tp, _Alloc >::operator=(), std::deque< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::reserve(), std::basic_string< _CharT, _Traits, _Alloc >::reserve(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize(), std::basic_string< _CharT, _Traits, _Alloc >::resize(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind(), and std::basic_string< _CharT, _Traits, _Alloc >::rfind().

3.11.4.286 ios_base& std::skipws (ios_base & __base) [inline]

Calls base.setf(ios_base::skipws).

Definition at line 859 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::skipws.

Referenced by operator>>().

3.11.4.287 template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::static_pointer_cast (const __shared_ptr<_Tp1, _Lp> & __r) [inline], [noexcept]

static_pointer_cast

Definition at line 1186 of file shared_ptr_base.h.

3.11.4.288 template<typename _Key, typename _Compare, typename _Alloc> void std::swap (multiset< _Key, _Compare, _Alloc> & __x, multiset< _Key, _Compare, _Alloc> & __y) [inline]

See std::multiset::swap().

Definition at line 789 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc>::swap().

3.11.4.289 template<typename _Key, typename _Compare, typename _Alloc> void std::swap (set< _Key, _Compare, _Alloc> & __x, set< _Key, _Compare, _Alloc> & __y) [inline]

See std::set::swap().

Definition at line 804 of file stl_set.h.

References std::set< _Key, _Compare, _Alloc>::swap().

3.11.4.290 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void std::swap (multimap< _Key, _Tp, _Compare, _Alloc> & __x, multimap< _Key, _Tp, _Compare, _Alloc> & __y) [inline]

See std::multimap::swap().

Definition at line 915 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc>::swap().

3.11.4.291 template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> void std::swap (map< _Key, _Tp, _Compare, _Alloc> & __x, map< _Key, _Tp, _Compare, _Alloc> & __y) [inline]

See std::map::swap().

Definition at line 1013 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc>::swap().

3.11.4.292 `template<typename _Tp, typename _Alloc> void std::swap (forward_list< _Tp, _Alloc> & __x, forward_list< _Tp, _Alloc> & __y) [inline]`

See `std::forward_list::swap()`.

Definition at line 1395 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc>::swap()`.

3.11.4.293 `template<typename _Tp, typename _Alloc> void std::swap (vector< _Tp, _Alloc> & __x, vector< _Tp, _Alloc> & __y) [inline]`

See `std::vector::swap()`.

Definition at line 1447 of file `stl_vector.h`.

References `std::vector< _Tp, _Alloc>::swap()`.

3.11.4.294 `template<typename _Tp, typename _Alloc> void std::swap (list< _Tp, _Alloc> & __x, list< _Tp, _Alloc> & __y) [inline]`

See `std::list::swap()`.

Definition at line 1658 of file `stl_list.h`.

References `std::list< _Tp, _Alloc>::swap()`.

3.11.4.295 `template<typename _Tp, typename _Alloc> void std::swap (deque< _Tp, _Alloc> & __x, deque< _Tp, _Alloc> & __y) [inline]`

See `std::deque::swap()`.

Definition at line 2011 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc>::swap()`.

3.11.4.296 `template<typename _Res, typename... _Args> void std::swap (function< _Res(_Args...)> & __x, function< _Res(_Args...)> & __y) [inline]`

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 2570 of file `functional`.

3.11.4.297 `template<typename _CharT, typename _Traits, typename _Alloc> void std::swap (basic_string< _CharT, _Traits, _Alloc> & __lhs, basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Swap contents of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2717 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::swap()`.

3.11.4.298 `bool std::test (size_t __position) const`

Tests the value of a bit.

Parameters

<code>__position</code>	The index of a bit.
-------------------------	---------------------

Returns

The value at *pos*.

Exceptions

<code>std::out_of_range</code>	If <i>pos</i> is bigger the size of the set.
--------------------------------	--

Definition at line 1303 of file `bitset`.

References `_Unchecked_test()`.

3.11.4.299 `template<class _CharT , class _Traits , class _Alloc > std::basic_string< char, std::char_traits< char >, std::allocator< char > > std::to_string () const`

Returns a character interpretation of the bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1179 of file `bitset`.

3.11.4.300 `unsigned long std::to_ulong () const`

Returns a numerical interpretation of the bitset.

Returns

The integral equivalent of the bits.

Exceptions

<code>std::overflow_error</code>	If there are too many bits to be represented in an <code>unsigned long</code> .
----------------------------------	---

Definition at line 1160 of file `bitset`.

3.11.4.301 `template<typename _CharT > _CharT std::tolower (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype::tolower(__c)`.

Definition at line 2602 of file `locale_facets.h`.

3.11.4.302 `template<typename _CharT > _CharT std::toupper (_CharT __c, const locale & __loc) [inline]`

Convenience interface to ctype.toupper(__c).

Definition at line 2596 of file locale_facets.h.

3.11.4.303 `template<typename _InputIterator , typename _ForwardIterator > _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result) [inline]`

Copies the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`__result + (__first - __last)`

Like copy(), but does not require an initialized output range.

Definition at line 107 of file stl_uninitialized.h.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

3.11.4.304 `template<typename _InputIterator , typename _Size , typename _ForwardIterator > _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result) [inline]`

Copies the range [first,first+n) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

`__result + __n`

Like copy_n(), but does not require an initialized output range.

Definition at line 647 of file stl_uninitialized.h.

References `__iterator_category()`.

3.11.4.305 `template<typename _ForwardIterator , typename _Tp > void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x) [inline]`

Copies the value x into the range [first,last).

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	The source value.

Returns

Nothing.

Like fill(), but does not require an initialized output range.

Definition at line 164 of file stl_uninitialized.h.

3.11.4.306 `template<typename _ForwardIterator, typename _Size, typename _Tp> void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp & __x) [inline]`

Copies the value x into the range [first,first+n).

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of copies to make.
<code>__x</code>	The source value.

Returns

Nothing.

Like fill_n(), but does not require an initialized output range.

Definition at line 218 of file stl_uninitialized.h.

3.11.4.307 `ios_base& std::unitbuf (ios_base & __base) [inline]`

Calls base.setf(ios_base::unitbuf).

Definition at line 891 of file ios_base.h.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::unitbuf`.

3.11.4.308 `ios_base& std::uppercase (ios_base & __base) [inline]`

Calls base.setf(ios_base::uppercase).

Definition at line 875 of file ios_base.h.

References `__gnu_debug::__base()`, `std::ios_base::setf()`, and `std::ios_base::uppercase`.

3.11.4.309 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::ws (basic_istream< _CharT, _Traits> & __is)`

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;
std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a `std::basic_istream::sentry` inside your definition of operator>>.

Definition at line 1016 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

3.11.5 Variable Documentation

3.11.5.1 `ios_base::Init` `std::__ioinit` [static]

Linked to standard error (buffered)

Definition at line 74 of file iostream.

3.11.5.2 `ostream` `std::cerr`

Linked to standard output.

3.11.5.3 `istream` `std::cin`

Linked to standard input.

3.11.5.4 `ostream` `std::clog`

Linked to standard error (unbuffered)

3.11.5.5 `ostream` `std::cout`

Linked to standard input.

3.11.5.6 `wostream` `std::wcerr`

Linked to standard output.

3.11.5.7 `wistream` `std::wcin`

Linked to standard error (buffered)

3.11.5.8 `wostream` `std::wclog`

Linked to standard error (unbuffered)

3.11.5.9 `wostream` `std::wcout`

Linked to standard input.

3.12 std::__debug Namespace Reference

Classes

- class [bitset](#)
Class `std::bitset` with additional safety/checking/debug instrumentation.
- class [deque](#)
Class `std::deque` with safety/checking/debug instrumentation.
- class [forward_list](#)
Class `std::forward_list` with safety/checking/debug instrumentation.

- class [list](#)
Class std::list with safety/checking/debug instrumentation.
- class [map](#)
Class std::map with safety/checking/debug instrumentation.
- class [multimap](#)
Class std::multimap with safety/checking/debug instrumentation.
- class [multiset](#)
Class std::multiset with safety/checking/debug instrumentation.
- class [set](#)
Class std::set with safety/checking/debug instrumentation.
- class [unordered_map](#)
Class std::unordered_map with safety/checking/debug instrumentation.
- class [unordered_multimap](#)
Class std::unordered_multimap with safety/checking/debug instrumentation.
- class [unordered_multiset](#)
Class std::unordered_multiset with safety/checking/debug instrumentation.
- class [unordered_set](#)
Class std::unordered_set with safety/checking/debug instrumentation.
- class [vector](#)
Class std::vector with safety/checking/debug instrumentation.

Functions

- template<typename _Key, typename _Compare, typename _Allocator >
bool **operator!=** (const [multiset](#)< _Key, _Compare, _Allocator > &__lhs, const [multiset](#)< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool **operator!=** (const [multimap](#)< _Key, _Tp, _Compare, _Allocator > &__lhs, const [multimap](#)< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator >
bool **operator!=** (const [set](#)< _Key, _Compare, _Allocator > &__lhs, const [set](#)< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool **operator!=** (const [map](#)< _Key, _Tp, _Compare, _Allocator > &__lhs, const [map](#)< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool **operator!=** (const [unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__x, const [unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
bool **operator!=** (const [unordered_map](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const [unordered_map](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp, typename _Alloc >
bool **operator!=** (const [deque](#)< _Tp, _Alloc > &__lhs, const [deque](#)< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator!=** (const [vector](#)< _Tp, _Alloc > &__lhs, const [vector](#)< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator!=** (const [list](#)< _Tp, _Alloc > &__lhs, const [list](#)< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator!=** (const [forward_list](#)< _Tp, _Alloc > &__lx, const [forward_list](#)< _Tp, _Alloc > &__ly)

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _`
`Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,`
`_Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value,`
`_Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map<`
`_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset<`
`_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_`
`multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _`
`Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,`
`_Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > &operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

3.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior. Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

3.12.2 Function Documentation

- 3.12.2.1 `template<typename _Tp, typename _Alloc > bool std::__debug::operator<= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 768 of file debug/forward_list.

- 3.12.2.2 `template<typename _Tp, typename _Alloc > bool std::__debug::operator> (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 754 of file debug/forward_list.

- 3.12.2.3 `template<typename _Tp, typename _Alloc > bool std::__debug::operator>= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 761 of file debug/forward_list.

- 3.12.2.4 `template<typename _Tp, typename _Alloc > void std::__debug::swap (forward_list< _Tp, _Alloc > & __lx, forward_list< _Tp, _Alloc > & __ly) [inline]`

See `std::forward_list::swap()`.

Definition at line 775 of file debug/forward_list.

3.13 std::__detail Namespace Reference

Classes

- class [_Automaton](#)
Base class for, um, automata. Could be an NFA or a DFA. Your choice.
- struct [_Before_begin](#)
- struct [_CharMatcher](#)
Matches a single character.
- class [_Compiler](#)
Builds an NFA from an input iterator interval.
- struct [_Default_ranged_hash](#)

Default ranged hash function H . In principle it should be a function object composed from objects of type $H1$ and $H2$ such that $h(k, N) = h2(h1(k), N)$, but that would mean making extra copies of $h1$ and $h2$. So instead we'll just use a tag to tell class template hashtable to do that composition.

- struct [_EndTagger](#)
End state tag.
- struct [_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >](#)
Specialization.
- struct [_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >](#)
Specialization.
- struct [_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >](#)
Specialization.
- struct [_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >](#)
Specialization.
- struct [_Equality_base](#)
- class [_Grep_matcher](#)
Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.
- struct [_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >](#)
Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.
- struct [_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >](#)
Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.
- struct [_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >](#)
Specialization: ranged hash function, no caching hash codes. $H1$ and $H2$ are provided but ignored. We define a dummy hash code type.
- struct [_Hash_node< _Value, false >](#)
- struct [_Hash_node< _Value, true >](#)
- struct [_Hash_node_base](#)
- struct [_Hashtable_base](#)
- struct [_Hashtable_ebo_helper< _Nm, _Tp, false >](#)
Specialization not using EBO.
- struct [_Hashtable_ebo_helper< _Nm, _Tp, true >](#)
Specialization using EBO.
- struct [_Hashtable_traits](#)
- struct [_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >](#)
Specialization.
- struct [_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >](#)
Specialization.
- struct [_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >](#)
Specialization.
- struct [_Insert_base](#)
- struct [_List_node_base](#)
Common part of a node in the list.
- struct [_Local_const_iterator](#)
local const iterators

- struct [_Local_iterator](#)
local iterators
- struct [_Local_iterator_base](#)< [_Key](#), [_Value](#), [_ExtractKey](#), [_H1](#), [_H2](#), [_Hash](#), false >
Specialization.
- struct [_Local_iterator_base](#)< [_Key](#), [_Value](#), [_ExtractKey](#), [_H1](#), [_H2](#), [_Hash](#), true >
Specialization.
- struct [_Map_base](#)
- struct [_Map_base](#)< [_Key](#), [_Pair](#), [_Alloc](#), [_Select1st](#), [_Equal](#), [_H1](#), [_H2](#), [_Hash](#), [_RehashPolicy](#), [_Traits](#), false >
Partial specialization, __unique_keys set to false.
- struct [_Map_base](#)< [_Key](#), [_Pair](#), [_Alloc](#), [_Select1st](#), [_Equal](#), [_H1](#), [_H2](#), [_Hash](#), [_RehashPolicy](#), [_Traits](#), true >
Partial specialization, __unique_keys set to true.
- struct [_Mod_range_hashing](#)
Default range hashing function: use division to fold a large number into the range [0, N).
- class [_Nfa](#)
struct _Nfa
- struct [_Node_const_iterator](#)
Node const_iterators, used to iterate through all the hashtable.
- struct [_Node_iterator](#)
Node iterators, used to iterate through all the hashtable.
- struct [_Node_iterator_base](#)
Base class for node iterators.
- struct [_PatternCursor](#)
ABC for pattern matching.
- struct [_Prime_rehash_policy](#)
Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.
- struct [_RangeMatcher](#)
Matches a character range (bracket expression)
- struct [_Rehash_base](#)< [_Key](#), [_Value](#), [_Alloc](#), [_ExtractKey](#), [_Equal](#), [_H1](#), [_H2](#), [_Hash](#), [_Prime_rehash_policy](#), [_Traits](#) >
Specialization.
- struct [_Results](#)
Provides a generic facade for a templated match_results.
- class [_Scanner](#)
struct _Scanner. Scans an input range for regex tokens.
- struct [_Scanner_base](#)
Base class for scanner.
- class [_SpecializedCursor](#)
Provides a cursor into the specific target string.
- class [_SpecializedResults](#)
A _Results facade specialized for wrapping a templated match_results.
- struct [_StartTagger](#)
Start state tag.
- struct [_State](#)
struct _State
- class [_StateSeq](#)
Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Typedefs

- typedef [std::shared_ptr](#)
 < [_Automaton](#) > [_AutomatonPtr](#)
- typedef std::function< bool(const
 [_PatternCursor](#) &)> [_Matcher](#)
- typedef int [_StateldT](#)
- typedef [std::set](#)< [_StateldT](#) > [_StateSet](#)
- typedef [std::stack](#)< [_StateldT](#),
 [std::vector](#)< [_StateldT](#) > > [_StateStack](#)
- typedef std::function< void(const
 [_PatternCursor](#) &, [_Results](#) &)> [_Tagger](#)

Enumerations

- enum [_Opcode](#) {
 [_S_opcode_unknown](#), [_S_opcode_alternative](#), [_S_opcode_subexpr_begin](#), [_S_opcode_subexpr_end](#),
 [_S_opcode_match](#), [_S_opcode_accept](#) }

Functions

- template<typename [_InIter](#) , typename [_TraitsT](#) >
 [_AutomatonPtr](#) [__compile](#) (const [_InIter](#) &__b, const [_InIter](#) &__e, [_TraitsT](#) &__t, [regex_constants::syntax_option_type](#) __f)
- template<typename [_FwdIterT](#) >
 [_SpecializedCursor](#)< [_FwdIterT](#) > [__cursor](#) (const [_FwdIterT](#) &__b, const [_FwdIterT](#) __e)
- template<class [_Iterator](#) >
 std::iterator_traits
 < [_Iterator](#) >::difference_type [__distance_fw](#) ([_Iterator](#) __first, [_Iterator](#) __last, [std::input_iterator_tag](#))
- template<class [_Iterator](#) >
 std::iterator_traits
 < [_Iterator](#) >::difference_type [__distance_fw](#) ([_Iterator](#) __first, [_Iterator](#) __last, [std::forward_iterator_tag](#))
- template<class [_Iterator](#) >
 std::iterator_traits
 < [_Iterator](#) >::difference_type [__distance_fw](#) ([_Iterator](#) __first, [_Iterator](#) __last)
- template<typename [_InputIterator](#) , typename [_OutputIterator](#) , typename [_Tp](#) >
 [_OutputIterator](#) [__normalize](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [_OutputIterator](#) __result, const [_Tp](#) &__factor)
- bool [_AnyMatcher](#) (const [_PatternCursor](#) &)
- template<typename [_Value](#) , bool [_Cache_hash_code](#)>
 bool [operator!=](#) (const [_Node_iterator_base](#)< [_Value](#), [_Cache_hash_code](#) > &__x, const [_Node_iterator_base](#)< [_Value](#), [_Cache_hash_code](#) > &__y)
- template<typename [_Key](#) , typename [_Value](#) , typename [_ExtractKey](#) , typename [_H1](#) , typename [_H2](#) , typename [_Hash](#) , bool [__cache](#)>
 bool [operator!=](#) (const [_Local_iterator_base](#)< [_Key](#), [_Value](#), [_ExtractKey](#), [_H1](#), [_H2](#), [_Hash](#), [__cache](#) > &__x, const [_Local_iterator_base](#)< [_Key](#), [_Value](#), [_ExtractKey](#), [_H1](#), [_H2](#), [_Hash](#), [__cache](#) > &__y)
- template<typename [_Value](#) , bool [_Cache_hash_code](#)>
 bool [operator==](#) (const [_Node_iterator_base](#)< [_Value](#), [_Cache_hash_code](#) > &__x, const [_Node_iterator_base](#)< [_Value](#), [_Cache_hash_code](#) > &__y)
- template<typename [_Key](#) , typename [_Value](#) , typename [_ExtractKey](#) , typename [_H1](#) , typename [_H2](#) , typename [_Hash](#) , bool [__cache](#)>
 bool [operator==](#) (const [_Local_iterator_base](#)< [_Key](#), [_Value](#), [_ExtractKey](#), [_H1](#), [_H2](#), [_Hash](#), [__cache](#) > &__x, const [_Local_iterator_base](#)< [_Key](#), [_Value](#), [_ExtractKey](#), [_H1](#), [_H2](#), [_Hash](#), [__cache](#) > &__y)

Variables

- static const [_StateIdT _S_invalid_state_id](#)

3.13.1 Detailed Description

Implementation details not part of the namespace std interface.

3.14 std::__parallel Namespace Reference

Classes

- struct [_CRandNumber](#)
Functor wrapper for std::rand().

Functions

- template<typename _Iter, typename _Tp, typename _Tag >
_Tp **__accumulate_switch** (_Iter, _Iter, _Tp, _Tag)
- template<typename _Iter, typename _Tp, typename _IteratorTag >
_Tp **__accumulate_switch** (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)
- template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >
_Tp **__accumulate_switch** (_Iter, _Iter, _Tp, _BinaryOper, _Tag)
- template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >
_Tp **__accumulate_switch** (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)
- template<typename _RAIter, typename _Tp, typename _BinaryOper >
_Tp **__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) __parallelism=[__gnu_parallel::parallel_unbalanced](#))
- template<typename __RAIter, typename _Tp, typename _BinaryOperation >
_Tp **__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) __parallelism_tag=[__gnu_parallel::parallel_unbalanced](#))
- template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >
_OIter **__adjacent_difference_switch** (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter **__adjacent_difference_switch** (_Iter, _Iter, _OIter, _BinaryOper, [random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) __parallelism=[__gnu_parallel::parallel_unbalanced](#))
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >
_OutputIterator **__adjacent_difference_switch** (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **__adjacent_difference_switch** (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, [random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) __parallelism_tag=[__gnu_parallel::parallel_balanced](#))
- template<typename _Filter, typename _IterTag >
_Filter **__adjacent_find_switch** (_Filter, _Filter, _IterTag)
- template<typename _Filter, typename _BiPredicate, typename _IterTag >
_Filter **__adjacent_find_switch** (_Filter, _Filter, _BiPredicate, _IterTag)
- template<typename _RAIter, typename _BiPredicate >
_RAIter **__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, [random_access_iterator_tag](#))

- `template<typename _RAIter >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _Filterator, typename _IteratorTag >`
`_Filterator __adjacent_find_switch (_Filterator __begin, _Filterator __end, _IteratorTag)`
- `template<typename _Filterator, typename _BinaryPredicate, typename _IteratorTag >`
`_Filterator __adjacent_find_switch (_Filterator __begin, _Filterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`iterator_traits< _Iter >`
`::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >`
`::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`iterator_traits< _Iter >`
`::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >`
`::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >`
`::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >`
`::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp & __value, _IteratorTag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Filterator, typename _IteratorTag1, typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _Filterator, typename _BinaryPredicate, typename _IteratorTag >`
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _Filterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`
`_Function __for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`
`_Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _Iter, typename _Function, typename _IterTag >`
`_Function __for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`
`_OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`
`_OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`
`void __generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _Filterator, typename _Generator, typename _IteratorTag >`
`void __generate_switch (_Filterator __begin, _Filterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`
`void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >`
`_Tp __inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp __inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Tp __inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`bool __lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)`

- `template<typename _Filterator, typename _Compare, typename _IteratorTag >`
`_Filterator __max_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter __merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator __merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag >`
`_Filterator __min_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`pair< _IIter1, _IIter2 > __mismatch_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`pair< _IIter1, _IIter2 > __mismatch_switch (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter __partial_sum_switch (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`
`_OIter __partial_sum_switch (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator __partial_sum_switch (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __partial_sum_switch (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`
`_Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag >`
`_Filterator __partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`
`void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`

- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IteratorTag >`
`void __replace_if_switch (_Filter __begin, _Filter __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`
`void __replace_switch (_Filter __begin, _Filter __end, const _Tp &__old_value, const _Tp &__new_value, _IterTag)`
- `template<typename _Filter, typename _Tp, typename _IteratorTag >`
`void __replace_switch (_Filter __begin, _Filter __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`
`_Filter __search_n_switch (_Filter __begin, _Filter __end, _Integer __count, const _Tp &__val, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`
`_Filter __search_n_switch (_Filter __begin, _Filter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`
`_Filter1 __search_switch (_Filter1 __begin1, _Filter1 __end1, _Filter2 __begin2, _Filter2 __end2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Filter1 __search_switch (_Filter1 __begin1, _Filter1 __end1, _Filter2 __begin2, _Filter2 __end2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Filter1 __search_switch (_Filter1 __begin1, _Filter1 __end1, _Filter2 __begin2, _Filter2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_Filter1 __search_switch (_Filter1 __begin1, _Filter1 __end1, _Filter2 __begin2, _Filter2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_Olter __set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`
`_Output_RAIter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_Olter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`
`_Output_RAIter __set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_Olter __set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`
`_Output_RAIter __set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_Olter __set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter, typename _Olter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`
`_Olter __transform1_switch (_Iter, _Iter, _Olter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOlter, typename _UnaryOperation >`
`_RAOlter __transform1_switch (_RAIter, _RAIter, _RAOlter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism=gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`

- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`
`_RAIter3 transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`
`_OIter transform2_switch (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`
`_RAIter3 transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`
`_OutputIterator transform2_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _IIter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator unique_copy_switch (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`
`RandomAccessOutputIterator unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`_OIter unique_copy_switch (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`
`_RandomAccess_OIter unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter, typename _Tp >`
`_Tp accumulate (_IIter, _IIter, _Tp)`
- `template<typename _IIter, typename _Tp >`
`_Tp accumulate (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _IIter, typename _Tp >`
`_Tp accumulate (_IIter, _IIter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation >`
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation >`
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation >`
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _IIter, typename _OIter >`
`_OIter adjacent_difference (_IIter, _IIter, _OIter)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_IIter, _IIter, _OIter, _BinaryOper)`
- `template<typename _IIter, typename _OIter >`
`_OIter adjacent_difference (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >`
`::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >`
`::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >`
`::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >`
`::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >`
`::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >`
`::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter, _Iter, _Function)`

- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _IIter1, typename _IIter2, typename _Tp >`
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp)`
- `template<typename _IIter1, typename _IIter2, typename _Tp >`
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Tp >`
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2 >`
`bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`
`bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2 >`
`bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`

- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2 >`
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2 >`
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`
`pair< _IIter1, _IIter2 > mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)`
- `template<typename _RAIter >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _IIter, typename _OIter >`
`_OIter partial_sum (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition (_Filter, _Filter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filterator, typename _Predicate >`
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate >`
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`
- `template<typename _RAlter >`
`void random_shuffle (_RAlter __begin, _RAlter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter >`
`void random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filterator, typename _Tp >`
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Tp >`
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp >`
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)`

- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::__Parallelism)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filterator1, typename _Filterator2 >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2 >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Filterator, typename _Integer, typename _Tp >`
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp >`
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _Iiter1, typename _Iiter2, typename _OutputIterator >`
`_OutputIterator set_difference (_Iiter1 __begin1, _Iiter1 __end1, _Iiter2 __begin2, _Iiter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output-`
`Iterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`

- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _Binary-
Operation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _Binary-
Operation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _Binary-
Operation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, __gnu_parallel::sequential-
_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, __gnu-
_parallel::sequential_tag)`

- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`

3.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

3.15 std::__profile Namespace Reference

Classes

- class [bitset](#)
Class std::bitset wrapper with performance instrumentation.
- class [deque](#)
Class std::deque wrapper with performance instrumentation.
- class [forward_list](#)
Class std::forward_list wrapper with performance instrumentation.
- class [list](#)
List wrapper with performance instrumentation.
- class [map](#)
Class std::map wrapper with performance instrumentation.
- class [multimap](#)
Class std::multimap wrapper with performance instrumentation.
- class [multiset](#)
Class std::multiset wrapper with performance instrumentation.
- class [set](#)
Class std::set wrapper with performance instrumentation.
- class [unordered_map](#)
Class std::unordered_map wrapper with performance instrumentation.
- class [unordered_multimap](#)
Class std::unordered_multimap wrapper with performance instrumentation.
- class [unordered_multiset](#)
Unordered_multiset wrapper with performance instrumentation.
- class [unordered_set](#)
Unordered_set wrapper with performance instrumentation.

Functions

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence > operator+ (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`__iterator_tracker< _IteratorL, _Sequence >::difference_type operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence >::difference_type operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

3.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

3.15.2 Function Documentation

3.15.2.1 `template<typename _Tp, typename _Alloc > bool std::__profile::operator<= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 166 of file profile/forward_list.

3.15.2.2 `template<typename _Tp, typename _Alloc > bool std::__profile::operator> (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 152 of file profile/forward_list.

3.15.2.3 `template<typename _Tp, typename _Alloc > bool std::__profile::operator>= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 159 of file profile/forward_list.

3.15.2.4 `template<typename _Tp, typename _Alloc > void std::__profile::swap (forward_list< _Tp, _Alloc > & __x, forward_list< _Tp, _Alloc > & __y) [inline]`

See `std::forward_list::swap()`.

Definition at line 173 of file `profile/forward_list`.

3.16 std::chrono Namespace Reference

Classes

- struct [duration](#)
duration
- struct [duration_values](#)
duration_values
- struct [system_clock](#)
system_clock
- struct [time_point](#)
time_point
- struct [treat_as_floating_point](#)
treat_as_floating_point

Typedefs

- typedef [system_clock](#) **high_resolution_clock**
- typedef [duration](#)< int, [ratio](#)< 3600 > > **hours**
- typedef [duration](#)< int64_t, [micro](#) > **microseconds**
- typedef [duration](#)< int64_t, [milli](#) > **milliseconds**
- typedef [duration](#)< int, [ratio](#)< 60 > > **minutes**
- typedef [duration](#)< int64_t, [nano](#) > **nanoseconds**
- typedef [duration](#)< int64_t > **seconds**
- typedef [system_clock](#) **steady_clock**

Functions

- `template<typename _ToDur, typename _Rep, typename _Period > constexpr enable_if < __is_duration< _ToDur > ::value, _ToDur >::type duration_cast (const duration< _Rep, _Period > & __d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 > constexpr bool operator!= (const duration< _Rep1, _Period1 > & __lhs, const duration< _Rep2, _Period2 > & __rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 > constexpr bool operator!= (const time_point< _Clock, _Dur1 > & __lhs, const time_point< _Clock, _Dur2 > & __rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 > constexpr duration< typename __common_rep_type< _Rep1, typename enable_if < !__is_duration< _Rep2 > ::value, _Rep2 >::type >::type, _Period > operator% (const duration< _Rep1, _Period > & __d, const _Rep2 & __s)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr common_type`
`< duration< _Rep1, _Period1 >`
`, duration< _Rep2, _Period2 >`
`>::type operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`constexpr duration< typename`
`__common_rep_type< _Rep1,`
`_Rep2 >::type, _Period > operator* (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Rep2, typename _Period >`
`constexpr duration< typename`
`__common_rep_type< _Rep2,`
`_Rep1 >::type, _Period > operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr common_type`
`< duration< _Rep1, _Period1 >`
`, duration< _Rep2, _Period2 >`
`>::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`
`constexpr time_point< _Clock,`
`typename common_type< _Dur1,`
`duration< _Rep2, _Period2 >`
`>::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >`
`constexpr time_point< _Clock,`
`typename common_type< duration`
`< _Rep1, _Period1 >, _Dur2 >`
`::type > operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr common_type`
`< duration< _Rep1, _Period1 >`
`, duration< _Rep2, _Period2 >`
`>::type operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`
`constexpr time_point< _Clock,`
`typename common_type< _Dur1,`
`duration< _Rep2, _Period2 >`
`>::type > operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr common_type< _Dur1,`
`_Dur2 >::type operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__`
`rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`constexpr duration< typename`
`__common_rep_type< _Rep1,`
`typename enable_if`
`<!__is_duration< _Rep2 >`
`::value, _Rep2 >::type >::type,`
`_Period > operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr common_type< _Rep1,`
`_Rep2 >::type operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`
`&__rhs)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr bool operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr bool operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr bool operator== (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr bool operator> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr bool operator>= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`
`&__rhs)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`
`constexpr enable_if`
`< __is_duration< _ToDur >`
`::value, time_point< _Clock,`
`_ToDur > >::type time_point_cast (const time_point< _Clock, _Dur > &__t)`

3.16.1 Detailed Description

ISO C++ 2011 entities sub-namespace for time and date.

3.16.2 Typedef Documentation

3.16.2.1 typedef duration<int, ratio<3600> > std::chrono::hours

hours

Definition at line 541 of file chrono.

3.16.2.2 typedef duration<int64_t, micro> std::chrono::microseconds

microseconds

Definition at line 529 of file chrono.

3.16.2.3 typedef duration<int64_t, milli> std::chrono::milliseconds

milliseconds

Definition at line 532 of file chrono.

3.16.2.4 typedef duration<int, ratio< 60> > std::chrono::minutes

minutes

Definition at line 538 of file chrono.

3.16.2.5 typedef duration<int64_t, nano> std::chrono::nanoseconds

nanoseconds

Definition at line 526 of file chrono.

3.16.2.6 typedef duration<int64_t> std::chrono::seconds

seconds

Definition at line 535 of file chrono.

3.16.3 Function Documentation**3.16.3.1 template<typename _ToDur, typename _Rep, typename _Period> constexpr enable_if<_is_duration<_ToDur>::value, _ToDur>::type std::chrono::duration_cast (const duration< _Rep, _Period> & __d)**

duration_cast

Definition at line 193 of file chrono.

Referenced by std::this_thread::sleep_for().

3.16.3.2 template<typename _ToDur, typename _Clock, typename _Dur> constexpr enable_if<_is_duration<_ToDur>::value, time_point<_Clock, _ToDur> >::type std::chrono::time_point_cast (const time_point< _Clock, _Dur> & __t)

time_point_cast

Definition at line 602 of file chrono.

3.17 std::decimal Namespace Reference**Classes**

- class [decimal128](#)
3.2.4 Class decimal128.
- class [decimal32](#)
3.2.2 Class decimal32.
- class [decimal64](#)
3.2.3 Class decimal64.

Functions

- double **decimal128_to_double** ([decimal128](#) __d)

- float **decimal128_to_float** ([decimal128](#) __d)
- long double **decimal128_to_long_double** ([decimal128](#) __d)
- long long **decimal128_to_long_long** ([decimal128](#) __d)
- double **decimal32_to_double** ([decimal32](#) __d)
- float **decimal32_to_float** ([decimal32](#) __d)
- long double **decimal32_to_long_double** ([decimal32](#) __d)
- long long **decimal32_to_long_long** ([decimal32](#) __d)
- double **decimal64_to_double** ([decimal64](#) __d)
- float **decimal64_to_float** ([decimal64](#) __d)
- long double **decimal64_to_long_double** ([decimal64](#) __d)
- long long **decimal64_to_long_long** ([decimal64](#) __d)
- double **decimal_to_double** ([decimal32](#) __d)
- double **decimal_to_double** ([decimal64](#) __d)
- double **decimal_to_double** ([decimal128](#) __d)
- float **decimal_to_float** ([decimal32](#) __d)
- float **decimal_to_float** ([decimal64](#) __d)
- float **decimal_to_float** ([decimal128](#) __d)
- long double **decimal_to_long_double** ([decimal32](#) __d)
- long double **decimal_to_long_double** ([decimal64](#) __d)
- long double **decimal_to_long_double** ([decimal128](#) __d)
- long long **decimal_to_long_long** ([decimal32](#) __d)
- long long **decimal_to_long_long** ([decimal64](#) __d)
- long long **decimal_to_long_long** ([decimal128](#) __d)
- static [decimal128](#) **make_decimal128** (long long __coeff, int __exp)
- static [decimal128](#) **make_decimal128** (unsigned long long __coeff, int __exp)
- static [decimal32](#) **make_decimal32** (long long __coeff, int __exp)
- static [decimal32](#) **make_decimal32** (unsigned long long __coeff, int __exp)
- static [decimal64](#) **make_decimal64** (long long __coeff, int __exp)
- static [decimal64](#) **make_decimal64** (unsigned long long __coeff, int __exp)
- bool **operator!=** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, int __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned int __rhs)
- bool **operator!=** ([decimal32](#) __lhs, long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, long long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, [decimal32](#) __rhs)
- bool **operator!=** (unsigned int __lhs, [decimal32](#) __rhs)
- bool **operator!=** (long __lhs, [decimal32](#) __rhs)
- bool **operator!=** (unsigned long __lhs, [decimal32](#) __rhs)
- bool **operator!=** (long long __lhs, [decimal32](#) __rhs)
- bool **operator!=** (unsigned long long __lhs, [decimal32](#) __rhs)
- bool **operator!=** ([decimal64](#) __lhs, [decimal32](#) __rhs)
- bool **operator!=** ([decimal64](#) __lhs, [decimal64](#) __rhs)
- bool **operator!=** ([decimal64](#) __lhs, [decimal128](#) __rhs)
- bool **operator!=** ([decimal64](#) __lhs, int __rhs)
- bool **operator!=** ([decimal64](#) __lhs, unsigned int __rhs)
- bool **operator!=** ([decimal64](#) __lhs, long __rhs)
- bool **operator!=** ([decimal64](#) __lhs, unsigned long __rhs)

- bool **operator!=** (decimal64 __lhs, long long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator!=** (long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator!=** (long long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal128 __lhs, int __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal128 __lhs, long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal128 __lhs, long long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator!=** (long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator!=** (long long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator*** (decimal32 __lhs, int __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator*** (int __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **operator*** (long __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **operator*** (long long __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator*** (decimal64 __lhs, int __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator*** (decimal64 __lhs, long __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator*** (decimal64 __lhs, long long __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator*** (int __lhs, decimal64 __rhs)
- decimal64 **operator*** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **operator*** (long __lhs, decimal64 __rhs)
- decimal64 **operator*** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **operator*** (long long __lhs, decimal64 __rhs)
- decimal64 **operator*** (unsigned long long __lhs, decimal64 __rhs)

- [decimal128 operator*](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, long __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator*](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (long __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (long long __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, int __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, long __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, long long __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [decimal32 operator+](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator+](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator+](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal128 operator+](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)

- [decimal128 operator+](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, long __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator+](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (long __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (long long __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- [decimal32 operator-](#) ([decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, int __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, long long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [decimal32 operator-](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator-](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, int __rhs)

- `decimal128 operator-` (`decimal128 __lhs`, unsigned int `__rhs`)
- `decimal128 operator-` (`decimal128 __lhs`, long `__rhs`)
- `decimal128 operator-` (`decimal128 __lhs`, unsigned long `__rhs`)
- `decimal128 operator-` (`decimal128 __lhs`, long long `__rhs`)
- `decimal128 operator-` (`decimal128 __lhs`, unsigned long long `__rhs`)
- `decimal128 operator-` (int `__lhs`, `decimal128 __rhs`)
- `decimal128 operator-` (unsigned int `__lhs`, `decimal128 __rhs`)
- `decimal128 operator-` (long `__lhs`, `decimal128 __rhs`)
- `decimal128 operator-` (unsigned long `__lhs`, `decimal128 __rhs`)
- `decimal128 operator-` (long long `__lhs`, `decimal128 __rhs`)
- `decimal128 operator-` (unsigned long long `__lhs`, `decimal128 __rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, `decimal32 __rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, int `__rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, unsigned int `__rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, long `__rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, unsigned long `__rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, long long `__rhs`)
- `decimal32 operator/` (`decimal32 __lhs`, unsigned long long `__rhs`)
- `decimal32 operator/` (int `__lhs`, `decimal32 __rhs`)
- `decimal32 operator/` (unsigned int `__lhs`, `decimal32 __rhs`)
- `decimal32 operator/` (long `__lhs`, `decimal32 __rhs`)
- `decimal32 operator/` (unsigned long `__lhs`, `decimal32 __rhs`)
- `decimal32 operator/` (long long `__lhs`, `decimal32 __rhs`)
- `decimal32 operator/` (unsigned long long `__lhs`, `decimal32 __rhs`)
- `decimal64 operator/` (`decimal32 __lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, `decimal32 __rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, int `__rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, unsigned int `__rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, long `__rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, unsigned long `__rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, long long `__rhs`)
- `decimal64 operator/` (`decimal64 __lhs`, unsigned long long `__rhs`)
- `decimal64 operator/` (int `__lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (unsigned int `__lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (long `__lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (unsigned long `__lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (long long `__lhs`, `decimal64 __rhs`)
- `decimal64 operator/` (unsigned long long `__lhs`, `decimal64 __rhs`)
- `decimal128 operator/` (`decimal32 __lhs`, `decimal128 __rhs`)
- `decimal128 operator/` (`decimal64 __lhs`, `decimal128 __rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, `decimal32 __rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, `decimal64 __rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, `decimal128 __rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, long `__rhs`)
- `decimal128 operator/` (long long `__lhs`, `decimal128 __rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, int `__rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, unsigned int `__rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, unsigned long `__rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, long long `__rhs`)
- `decimal128 operator/` (`decimal128 __lhs`, unsigned long long `__rhs`)

- [decimal128 operator/](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (long __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (unsigned long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, int __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, long __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, unsigned long __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, long long __rhs)
- bool [operator<](#) (int __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- bool [operator<](#) (long long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (unsigned int __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal32](#) __lhs, unsigned int __rhs)
- bool [operator<](#) (long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (unsigned long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (long long __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, unsigned int __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, int __rhs)
- bool [operator<](#) (int __lhs, [decimal64](#) __rhs)
- bool [operator<](#) (unsigned int __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, long long __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, long __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, unsigned long __rhs)
- bool [operator<](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- bool [operator<](#) (unsigned long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, unsigned int __rhs)
- bool [operator<](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- bool [operator<](#) (int __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (unsigned int __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (long long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) (long __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, unsigned long __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, int __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, long __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- bool [operator<](#) ([decimal128](#) __lhs, long long __rhs)
- bool [operator==](#) ([decimal32](#) __lhs, unsigned long __rhs)

- bool **operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **operator==** (decimal32 __lhs, int __rhs)
- bool **operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **operator==** (decimal32 __lhs, long __rhs)
- bool **operator==** (decimal32 __lhs, long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal32 __rhs)
- bool **operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **operator==** (long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **operator==** (long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator==** (long __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, long long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, int __rhs)
- bool **operator==** (decimal64 __lhs, long __rhs)
- bool **operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **operator==** (int __lhs, decimal128 __rhs)
- bool **operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **operator==** (long __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, long long __rhs)
- bool **operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, int __rhs)
- bool **operator==** (decimal128 __lhs, long __rhs)
- bool **operator>** (unsigned int __lhs, decimal32 __rhs)
- bool **operator>** (long __lhs, decimal32 __rhs)
- bool **operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>** (decimal32 __lhs, decimal32 __rhs)

- bool **operator**> (decimal32 __lhs, decimal64 __rhs)
- bool **operator**> (decimal32 __lhs, long __rhs)
- bool **operator**> (unsigned long __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**> (long long __lhs, decimal32 __rhs)
- bool **operator**> (decimal32 __lhs, unsigned int __rhs)
- bool **operator**> (int __lhs, decimal32 __rhs)
- bool **operator**> (decimal32 __lhs, int __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**> (decimal64 __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**> (long long __lhs, decimal64 __rhs)
- bool **operator**> (int __lhs, decimal64 __rhs)
- bool **operator**> (decimal64 __lhs, unsigned int __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long __rhs)
- bool **operator**> (decimal64 __lhs, decimal128 __rhs)
- bool **operator**> (decimal64 __lhs, long __rhs)
- bool **operator**> (decimal64 __lhs, long long __rhs)
- bool **operator**> (decimal64 __lhs, decimal64 __rhs)
- bool **operator**> (decimal64 __lhs, int __rhs)
- bool **operator**> (long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **operator**> (decimal128 __lhs, decimal128 __rhs)
- bool **operator**> (int __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long __rhs)
- bool **operator**> (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned int __rhs)
- bool **operator**> (unsigned int __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, decimal32 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**> (decimal128 __lhs, long __rhs)
- bool **operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, int __rhs)
- bool **operator**> (long long __lhs, decimal128 __rhs)
- bool **operator**> (long __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, decimal64 __rhs)
- bool **operator**> (decimal128 __lhs, long long __rhs)
- bool **operator**>= (long long __lhs, decimal32 __rhs)
- bool **operator**>= (unsigned long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal32 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal32 __lhs, decimal32 __rhs)
- bool **operator**>= (decimal32 __lhs, int __rhs)
- bool **operator**>= (decimal32 __lhs, decimal128 __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**>= (unsigned int __lhs, decimal32 __rhs)
- bool **operator**>= (long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal32 __lhs, long long __rhs)
- bool **operator**>= (int __lhs, decimal32 __rhs)

- bool **operator**>= (decimal32 __lhs, long __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned long __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal64 __lhs, long long __rhs)
- bool **operator**>= (decimal64 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal64 __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned long __rhs)
- bool **operator**>= (decimal64 __lhs, decimal128 __rhs)
- bool **operator**>= (long __lhs, decimal64 __rhs)
- bool **operator**>= (decimal64 __lhs, long __rhs)
- bool **operator**>= (unsigned int __lhs, decimal64 __rhs)
- bool **operator**>= (decimal64 __lhs, int __rhs)
- bool **operator**>= (unsigned long __lhs, decimal64 __rhs)
- bool **operator**>= (int __lhs, decimal64 __rhs)
- bool **operator**>= (long long __lhs, decimal64 __rhs)
- bool **operator**>= (decimal128 __lhs, int __rhs)
- bool **operator**>= (int __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned long __rhs)
- bool **operator**>= (long long __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal64 __rhs)
- bool **operator**>= (unsigned long __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal32 __rhs)
- bool **operator**>= (decimal128 __lhs, long __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal128 __lhs, long long __rhs)
- bool **operator**>= (decimal128 __lhs, decimal128 __rhs)
- bool **operator**>= (unsigned int __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**>= (long __lhs, decimal128 __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal128 __rhs)

3.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

3.17.2 Function Documentation

3.17.2.1 long long std::decimal::decimal32_to_long_long (decimal32 __d)

Non-conforming extension: Conversion to integral type.

3.18 std::placeholders Namespace Reference

Variables

- const [_Placeholder](#)< 1 > [_1](#)
- const [_Placeholder](#)< 10 > [_10](#)

- const [_Placeholder](#)< 11 > [_11](#)
- const [_Placeholder](#)< 12 > [_12](#)
- const [_Placeholder](#)< 13 > [_13](#)
- const [_Placeholder](#)< 14 > [_14](#)
- const [_Placeholder](#)< 15 > [_15](#)
- const [_Placeholder](#)< 16 > [_16](#)
- const [_Placeholder](#)< 17 > [_17](#)
- const [_Placeholder](#)< 18 > [_18](#)
- const [_Placeholder](#)< 19 > [_19](#)
- const [_Placeholder](#)< 2 > [_2](#)
- const [_Placeholder](#)< 20 > [_20](#)
- const [_Placeholder](#)< 21 > [_21](#)
- const [_Placeholder](#)< 22 > [_22](#)
- const [_Placeholder](#)< 23 > [_23](#)
- const [_Placeholder](#)< 24 > [_24](#)
- const [_Placeholder](#)< 25 > [_25](#)
- const [_Placeholder](#)< 26 > [_26](#)
- const [_Placeholder](#)< 27 > [_27](#)
- const [_Placeholder](#)< 28 > [_28](#)
- const [_Placeholder](#)< 29 > [_29](#)
- const [_Placeholder](#)< 3 > [_3](#)
- const [_Placeholder](#)< 4 > [_4](#)
- const [_Placeholder](#)< 5 > [_5](#)
- const [_Placeholder](#)< 6 > [_6](#)
- const [_Placeholder](#)< 7 > [_7](#)
- const [_Placeholder](#)< 8 > [_8](#)
- const [_Placeholder](#)< 9 > [_9](#)

3.18.1 Detailed Description

ISO C++11 entities sub-namespace for functional.

3.19 std::regex_constants Namespace Reference

5.1 Regular Expression Syntax Options

- enum [__syntax_option](#) {
[_S_ica](#)se, [_S_nosubs](#), [_S_optimize](#), [_S_collate](#),
[_S_ECMAScript](#), [_S_basic](#), [_S_extended](#), [_S_awk](#),
[_S_grep](#), [_S_egrep](#), [_S_syntax_last](#) }
- typedef unsigned int [syntax_option_type](#)
- constexpr [syntax_option_type](#) [ica](#)se
- constexpr [syntax_option_type](#) [nosubs](#)
- constexpr [syntax_option_type](#) [optimize](#)
- constexpr [syntax_option_type](#) [collate](#)
- constexpr [syntax_option_type](#) [ECMAScript](#)
- constexpr [syntax_option_type](#) [basic](#)
- constexpr [syntax_option_type](#) [extended](#)
- constexpr [syntax_option_type](#) [awk](#)
- constexpr [syntax_option_type](#) [grep](#)
- constexpr [syntax_option_type](#) [egrep](#)

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `__match_flag` {
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- typedef `std::bitset`
`< _S_match_flag_last >` `match_flag_type`
- constexpr `match_flag_type` `match_default`
- constexpr `match_flag_type` `match_not_bol`
- constexpr `match_flag_type` `match_not_eol`
- constexpr `match_flag_type` `match_not_bow`
- constexpr `match_flag_type` `match_not_eow`
- constexpr `match_flag_type` `match_any`
- constexpr `match_flag_type` `match_not_null`
- constexpr `match_flag_type` `match_continuous`
- constexpr `match_flag_type` `match_prev_avail`
- constexpr `match_flag_type` `format_default`
- constexpr `match_flag_type` `format_sed`
- constexpr `match_flag_type` `format_no_copy`
- constexpr `match_flag_type` `format_first_only`

5.3 Error Types

- enum `error_type` {
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_complexity`,
`_S_error_stack`, `_S_error_last` }
- constexpr `error_type` `error_collate` (`_S_error_collate`)
- constexpr `error_type` `error_ctype` (`_S_error_ctype`)
- constexpr `error_type` `error_escape` (`_S_error_escape`)
- constexpr `error_type` `error_backref` (`_S_error_backref`)
- constexpr `error_type` `error_brack` (`_S_error_brack`)
- constexpr `error_type` `error_paren` (`_S_error_paren`)
- constexpr `error_type` `error_brace` (`_S_error_brace`)
- constexpr `error_type` `error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type` `error_range` (`_S_error_range`)
- constexpr `error_type` `error_space` (`_S_error_space`)
- constexpr `error_type` `error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type` `error_complexity` (`_S_error_complexity`)
- constexpr `error_type` `error_stack` (`_S_error_stack`)

3.19.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

3.19.2 Typedef Documentation

3.19.2.1 typedef std::bitset<_S_match_flag_last> std::regex_constants::match_flag_type

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 198 of file `regex_constants.h`.

3.19.2.2 typedef unsigned int std::regex_constants::syntax_option_type

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 80 of file `regex_constants.h`.

3.19.3 Enumeration Type Documentation

3.19.3.1 enum std::regex_constants::__match_flag

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 175 of file `regex_constants.h`.

3.19.3.2 enum std::regex_constants::__syntax_option

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 54 of file `regex_constants.h`.

3.19.3.3 enum std::regex_constants::error_type

The expression contained an invalid collating element name.

Definition at line 49 of file `regex_error.h`.

3.19.4 Function Documentation

3.19.4.1 constexpr error_type std::regex_constants::error_backref (_S_error_backref)

The expression contained an invalid back reference.

3.19.4.2 `constexpr error_type std::regex_constants::error_badbrace (_S_error_badbrace)`

The expression contained an invalid range in a {} expression.

3.19.4.3 `constexpr error_type std::regex_constants::error_badrepeat (_S_error_badrepeat)`

One of *?+{ was not preceded by a valid regular expression.

3.19.4.4 `constexpr error_type std::regex_constants::error_brace (_S_error_brace)`

The expression contained mismatched { and }

3.19.4.5 `constexpr error_type std::regex_constants::error_brack (_S_error_brack)`

The expression contained mismatched [and].

3.19.4.6 `constexpr error_type std::regex_constants::error_collate (_S_error_collate)`

The expression contained an invalid collating element name.

3.19.4.7 `constexpr error_type std::regex_constants::error_complexity (_S_error_complexity)`

The complexity of an attempted match against a regular expression exceeded a pre-set level.

3.19.4.8 `constexpr error_type std::regex_constants::error_ctype (_S_error_ctype)`

The expression contained an invalid character class name.

3.19.4.9 `constexpr error_type std::regex_constants::error_escape (_S_error_escape)`

The expression contained an invalid escaped character, or a trailing escape.

3.19.4.10 `constexpr error_type std::regex_constants::error_paren (_S_error_paren)`

The expression contained mismatched (and).

3.19.4.11 `constexpr error_type std::regex_constants::error_range (_S_error_range)`

The expression contained an invalid character range, such as [b-a] in most encodings.

3.19.4.12 `constexpr error_type std::regex_constants::error_space (_S_error_space)`

There was insufficient memory to convert the expression into a finite state machine.

3.19.4.13 `constexpr error_type std::regex_constants::error_stack (_S_error_stack)`

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

3.19.5 Variable Documentation**3.19.5.1 `constexpr syntax_option_type std::regex_constants::awk`**

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\'`, `'`, and `\ddd` (where `ddd` is one, two, or three octal digits).

Definition at line 144 of file `regex_constants.h`.

3.19.5.2 constexpr syntax_option_type std::regex_constants::basic

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 126 of file `regex_constants.h`.

3.19.5.3 constexpr syntax_option_type std::regex_constants::collate

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

Definition at line 107 of file `regex_constants.h`.

3.19.5.4 constexpr syntax_option_type std::regex_constants::ECMAScript

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 117 of file `regex_constants.h`.

3.19.5.5 constexpr syntax_option_type std::regex_constants::egrep

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that newlines are treated as whitespace.

Definition at line 160 of file `regex_constants.h`.

3.19.5.6 constexpr syntax_option_type std::regex_constants::extended

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 134 of file `regex_constants.h`.

3.19.5.7 constexpr match_flag_type std::regex_constants::format_default

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.

- `$n` The *n*th capture, where *n* is in [1,9] and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on [01, 99]. If `nn <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Definition at line 280 of file `regex_constants.h`.

3.19.5.8 `constexpr match_flag_type std::regex_constants::format_first_only`

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 301 of file `regex_constants.h`.

3.19.5.9 `constexpr match_flag_type std::regex_constants::format_no_copy`

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 295 of file `regex_constants.h`.

3.19.5.10 `constexpr match_flag_type std::regex_constants::format_sed`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1-2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 288 of file `regex_constants.h`.

3.19.5.11 `constexpr syntax_option_type std::regex_constants::grep`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

Definition at line 152 of file `regex_constants.h`.

3.19.5.12 `constexpr syntax_option_type std::regex_constants::icase`

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 86 of file `regex_constants.h`.

3.19.5.13 `constexpr match_flag_type std::regex_constants::match_any`

If more than one match is possible then any match is an acceptable result.

Definition at line 235 of file `regex_constants.h`.

3.19.5.14 `constexpr match_flag_type std::regex_constants::match_continuous`

The expression only matches a sub-sequence that begins at first .

Definition at line 245 of file `regex_constants.h`.

3.19.5.15 `constexpr match_flag_type std::regex_constants::match_default`

The default matching rules.

Definition at line 203 of file `regex_constants.h`.

3.19.5.16 `constexpr match_flag_type std::regex_constants::match_not_bol`

The first character in the sequence `[first, last)` is treated as though it is not at the beginning of a line, so the character `(^)` in the regular expression shall not match `[first, first)`.

Definition at line 210 of file `regex_constants.h`.

3.19.5.17 `constexpr match_flag_type std::regex_constants::match_not_bow`

The expression `\b` is not matched against the sub-sequence `[first,first)`.

Definition at line 223 of file `regex_constants.h`.

3.19.5.18 `constexpr match_flag_type std::regex_constants::match_not_eol`

The last character in the sequence `[first, last)` is treated as though it is not at the end of a line, so the character `($)` in the regular expression shall not match `[last, last)`.

Definition at line 217 of file `regex_constants.h`.

3.19.5.19 `constexpr match_flag_type std::regex_constants::match_not_eow`

The expression `\b` should not be matched against the sub-sequence `[last,last)`.

Definition at line 229 of file `regex_constants.h`.

3.19.5.20 `constexpr match_flag_type std::regex_constants::match_not_null`

The expression does not match an empty sequence.

Definition at line 240 of file `regex_constants.h`.

3.19.5.21 `constexpr match_flag_type std::regex_constants::match_prev_avail`

`—first` is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the regular expression algorithms 28.11 and iterators 28.12.

Definition at line 252 of file `regex_constants.h`.

3.19.5.22 `constexpr syntax_option_type std::regex_constants::nosubs`

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied `match_results` structure.

Definition at line 93 of file `regex_constants.h`.

3.19.5.23 `constexpr syntax_option_type std::regex_constants::optimize`

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 101 of file `regex_constants.h`.

3.20 `std::rel_ops` Namespace Reference

Functions

- `template<class _Tp >`
`bool operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`
`bool operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`
`bool operator>= (const _Tp &__x, const _Tp &__y)`

3.20.1 Detailed Description

The generated relational operators are sequestered here.

3.20.2 Function Documentation

3.20.2.1 `template<class _Tp > bool std::rel_ops::operator!= (const _Tp & __x, const _Tp & __y) [inline]`

Defines != for arbitrary types, in terms of ==.

Parameters

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

Returns

`__x != __y`

This function uses == to determine its result.

Definition at line 87 of file `stl_relops.h`.

3.20.2.2 `template<class _Tp > bool std::rel_ops::operator<= (const _Tp & __x, const _Tp & __y) [inline]`

Defines <= for arbitrary types, in terms of <.

Parameters

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

Returns

`__x <= __y`

This function uses < to determine its result.

Definition at line 113 of file `stl_relops.h`.

3.20.2.3 `template<class _Tp > bool std::rel_ops::operator> (const _Tp & __x, const _Tp & __y) [inline]`

Defines > for arbitrary types, in terms of <.

Parameters

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

Returns

`__x > __y`

This function uses `<` to determine its result.

Definition at line 100 of file `stl_relops.h`.

3.20.2.4 `template<class _Tp> bool std::rel_ops::operator>= (const _Tp & __x, const _Tp & __y) [inline]`

Defines `>=` for arbitrary types, in terms of `<`.

Parameters

<code>__x</code>	A thing.
<code>__y</code>	Another thing.

Returns

`__x >= __y`

This function uses `<` to determine its result.

Definition at line 126 of file `stl_relops.h`.

3.21 std::this_thread Namespace Reference

Functions

- void `__sleep_for` ([chrono::seconds](#), [chrono::nanoseconds](#))
- [thread::id](#) `get_id` () noexcept
- template<typename _Rep , typename _Period >
void `sleep_for` (const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Clock , typename _Duration >
void `sleep_until` (const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- void [yield](#) () noexcept

3.21.1 Detailed Description

ISO C++ 2011 entities sub-namespace for thread. 30.3.2 Namespace `this_thread`.

3.21.2 Function Documentation

3.21.2.1 `thread::id` `std::this_thread::get_id` () [inline], [noexcept]

`get_id`

Definition at line 252 of file `thread`.

3.21.2.2 `template<typename _Rep, typename _Period> void std::this_thread::sleep_for (const chrono::duration< _Rep, _Period> & _rtime) [inline]`

sleep_for

Definition at line 269 of file thread.

References `std::chrono::duration_cast()`.

Referenced by `sleep_until()`.

3.21.2.3 `template<typename _Clock, typename _Duration> void std::this_thread::sleep_until (const chrono::time_point< _Clock, _Duration> & _atime) [inline]`

sleep_until

Definition at line 288 of file thread.

References `sleep_for()`.

3.21.2.4 `void std::this_thread::yield () [inline], [noexcept]`

yield

Definition at line 256 of file thread.

3.22 std::tr1 Namespace Reference

Namespaces

- namespace [__detail](#)

Functions

- `template<typename _Tp> std::complex< _Tp> __complex_acosh (const std::complex< _Tp> & __z)`
- `template<typename _Tp> std::complex< _Tp> __complex_asinh (const std::complex< _Tp> & __z)`
- `template<typename _Tp> std::complex< _Tp> __complex_atanh (const std::complex< _Tp> & __z)`
- `template<typename _Tp> std::complex< _Tp> acosh (const std::complex< _Tp> & __z)`
- `template<typename _Tp> std::complex< _Tp> asinh (const std::complex< _Tp> & __z)`
- `template<typename _Tp> __gnu_cxx::__promote< _Tp>::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp> __gnu_cxx::__promote< _Tp>::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tp> std::complex< _Tp> atanh (const std::complex< _Tp> & __z)`

- `template<typename _Tpx, typename _Tpy >`
`__gnu_cxx::__promote_2< _Tpx,`
`_Tpy >::__type beta (_Tpx __x, _Tpy __y)`
- `float betaf (float __x, float __y)`
- `long double betal (long double __x, long double __y)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)`
- `float comp_ellint_1f (float __k)`
- `long double comp_ellint_1l (long double __k)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)`
- `float comp_ellint_2f (float __k)`
- `long double comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn >`
`__gnu_cxx::__promote_2< _Tp,`
`_Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float comp_ellint_3f (float __k, float __nu)`
- `long double comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa,`
`_Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hyperg (float __a, float __c, float __x)`
- `long double conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tp >`
`std::complex< _Tp > conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< typename`
`__gnu_cxx::__promote< _Tp >`
`::__type > conj (_Tp __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu,`
`_Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu,`
`_Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu,`
`_Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu,`
`_Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp,`
`_Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)`

- float **ellint_1f** (float __k, float __phi)
- long double **ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp >
__gnu_cxx::__promote_2< _Tp,
_Tpp >::__type **ellint_2** (_Tp __k, _Tpp __phi)
- float **ellint_2f** (float __k, float __phi)
- long double **ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp >
__gnu_cxx::__promote_3< _Tp,
_Tpn, _Tpp >::__type **ellint_3** (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **ellint_3f** (float __k, float __nu, float __phi)
- long double **ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **expint** (_Tp __x)
- float **expintf** (float __x)
- long double **expintl** (long double __x)
- template<typename _Tp >
std::complex< _Tp > **fabs** (const **std::complex**< _Tp > &__z)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **hermite** (unsigned int __n, _Tp __x)
- float **hermitef** (unsigned int __n, float __x)
- long double **hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_4< _Tpa,
_Tpb, _Tpc, _Tp >::__type **hyperg** (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **hypergff** (float __a, float __b, float __c, float __x)
- long double **hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **laguerre** (unsigned int __n, _Tp __x)
- float **laguerref** (unsigned int __n, float __x)
- long double **laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **legendre** (unsigned int __n, _Tp __x)
- float **legendref** (unsigned int __n, float __x)
- long double **legendrel** (unsigned int __n, long double __x)
- template<typename _Tp, typename _Up >
std::complex< typename
__gnu_cxx::__promote_2< _Tp,
_Up >::__type > **polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp, typename _Up >
std::complex< typename
__gnu_cxx::__promote_2< _Tp,
_Up >::__type > **pow** (const **std::complex**< _Tp > &__x, const _Up &__y)
- template<typename _Tp, typename _Up >
std::complex< typename
__gnu_cxx::__promote_2< _Tp,
_Up >::__type > **pow** (const _Tp &__x, const **std::complex**< _Up > &__y)
- template<typename _Tp, typename _Up >
std::complex< typename
__gnu_cxx::__promote_2< _Tp,
_Up >::__type > **pow** (const **std::complex**< _Tp > &__x, const **std::complex**< _Up > &__y)

- `template<typename _Tp >`
`std::complex< _Tp > pow` (`const std::complex< _Tp > &__x`, `const _Tp &__y`)
- `template<typename _Tp >`
`std::complex< _Tp > pow` (`const _Tp &__x`, `const std::complex< _Tp > &__y`)
- `template<typename _Tp >`
`std::complex< _Tp > pow` (`const std::complex< _Tp > &__x`, `const std::complex< _Tp > &__y`)
- `double pow` (`double __x`, `double __y`)
- `float pow` (`float __x`, `float __y`)
- `long double pow` (`long double __x`, `long double __y`)
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< _Tp,`
`_Up >::__type pow` (`_Tp __x`, `_Up __y`)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta` (`_Tp __x`)
- `float riemann_zetaf` (`float __x`)
- `long double riemann_zetal` (`long double __x`)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_bessel` (`unsigned int __n`, `_Tp __x`)
- `float sph_besself` (`unsigned int __n`, `float __x`)
- `long double sph_bessell` (`unsigned int __n`, `long double __x`)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_legendre` (`unsigned int __l`, `unsigned int __m`, `_Tp __theta`)
- `float sph_legendref` (`unsigned int __l`, `unsigned int __m`, `float __theta`)
- `long double sph_legendrel` (`unsigned int __l`, `unsigned int __m`, `long double __theta`)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_neumann` (`unsigned int __n`, `_Tp __x`)
- `float sph_neumannf` (`unsigned int __n`, `float __x`)
- `long double sph_neumannl` (`unsigned int __n`, `long double __x`)

3.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

3.23 std::tr1::__detail Namespace Reference

3.23.1 Detailed Description

Implementation details not part of the namespace std::tr1 interface.

3.24 std::tr2 Namespace Reference

Namespaces

- namespace [__detail](#)

Classes

- struct [__dynamic_bitset_base](#)
- struct [__reflection_typelist< _First, _Rest...>](#)
Partial specialization.
- struct [__reflection_typelist<>](#)
Specialization for an empty typelist.
- struct [bases](#)
Sequence abstraction metafunctions for manipulating a typelist.
- class [bool_set](#)
- struct [direct_bases](#)
Enumerate all the direct base classes of a class. Form of a typelist.
- class [dynamic_bitset](#)
The dynamic_bitset class represents a sequence of bits.

Functions

- bool **certainly** ([bool_set](#) __b)
- bool **contains** ([bool_set](#) __s, [bool_set](#) __t)
- bool **equals** ([bool_set](#) __s, [bool_set](#) __t)
- bool **is_emptyset** ([bool_set](#) __b)
- bool **is_indeterminate** ([bool_set](#) __b)
- bool **is_singleton** ([bool_set](#) __b)
- [bool_set](#) **operator!=** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **operator!=** ([bool_set](#) __s, [bool](#) __t)
- [bool_set](#) **operator!=** ([bool_set](#) __s, [bool_set](#) __t)
- [bool_set](#) **operator&** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **operator&** ([bool_set](#) __s, [bool](#) __t)
- [bool_set](#) **operator==** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **operator==** ([bool_set](#) __s, [bool](#) __t)
- [bool_set](#) **operator^** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **operator^** ([bool_set](#) __s, [bool](#) __t)
- [bool_set](#) **operator|** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **operator|** ([bool_set](#) __s, [bool](#) __t)
- bool **possibly** ([bool_set](#) __b)
- [bool_set](#) **set_complement** ([bool_set](#) __b)
- [bool_set](#) **set_intersection** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **set_intersection** ([bool_set](#) __s, [bool](#) __t)
- [bool_set](#) **set_intersection** ([bool_set](#) __s, [bool_set](#) __t)
- [bool_set](#) **set_union** ([bool](#) __s, [bool_set](#) __t)
- [bool_set](#) **set_union** ([bool_set](#) __s, [bool](#) __t)
- [bool_set](#) **set_union** ([bool_set](#) __s, [bool_set](#) __t)
- template<typename _WordT, typename _Alloc >
bool **operator==** (const [dynamic_bitset](#)< _WordT, _Alloc > &__lhs, const [dynamic_bitset](#)< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc >
bool **operator!=** (const [dynamic_bitset](#)< _WordT, _Alloc > &__lhs, const [dynamic_bitset](#)< _WordT, _Alloc > &__rhs)

- `template<typename _WordT, typename _Alloc >`
`bool operator< (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc >`
`&__rhs)`
- `template<typename _WordT, typename _Alloc >`
`bool operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc >`
`&__rhs)`
- `template<typename _WordT, typename _Alloc >`
`bool operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc >`
`&__rhs)`
- `template<typename _WordT, typename _Alloc >`
`bool operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc >`
`&__rhs)`
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_`
`bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_`
`bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_`
`bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_`
`bitset< _WordT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`
`std::basic_istream< _CharT,`
`_Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)`
- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`
`std::basic_ostream< _CharT,`
`_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc`
`> &__x)`

3.24.1 Detailed Description

ISO C++ TR2 entities toplevel namespace is std::tr2.

3.24.2 Function Documentation

- 3.24.2.1 `template<typename _WordT, typename _Alloc > bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &`
`__lhs, const dynamic_bitset< _WordT, _Alloc > & __rhs)`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1291 of file dynamic_bitset.

- 3.24.2.2 `template<typename _WordT, typename _Alloc > dynamic_bitset<_WordT, _Alloc> std::tr2::operator& (const`
`dynamic_bitset< _WordT, _Alloc > & __x, const dynamic_bitset< _WordT, _Alloc > & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1332 of file `dynamic_bitset`.

3.24.2.3 `template<typename _WordT, typename _Alloc > dynamic_bitset<_WordT, _Alloc> std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > & __x, const dynamic_bitset< _WordT, _Alloc > & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1362 of file `dynamic_bitset`.

3.24.2.4 `template<typename _WordT, typename _Alloc > bool std::tr2::operator< (const dynamic_bitset< _WordT, _Alloc > & __lhs, const dynamic_bitset< _WordT, _Alloc > & __rhs)`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1297 of file `dynamic_bitset`.

3.24.2.5 `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc > std::basic_ostream<_CharT, _Traits>& std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const dynamic_bitset< _WordT, _Alloc > & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace and only accept '0' and '1' characters. The `dynamic_bitset` will grow as necessary to hold the string of bits.

Definition at line 1455 of file `dynamic_bitset`.

References `std::ctype_abstract_base< _CharT >::widen()`.

3.24.2.6 `template<typename _WordT, typename _Alloc > bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > & __lhs, const dynamic_bitset< _WordT, _Alloc > & __rhs)`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1303 of file `dynamic_bitset`.

3.24.2.7 `template<typename _WordT, typename _Alloc> bool std::tr2::operator==(const dynamic_bitset< _WordT, _Alloc > & __lhs, const dynamic_bitset< _WordT, _Alloc > & __rhs)`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1285 of file `dynamic_bitset`.

3.24.2.8 `template<typename _WordT, typename _Alloc> bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > & __lhs, const dynamic_bitset< _WordT, _Alloc > & __rhs)`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1309 of file `dynamic_bitset`.

3.24.2.9 `template<typename _WordT, typename _Alloc> bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > & __lhs, const dynamic_bitset< _WordT, _Alloc > & __rhs)`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1315 of file `dynamic_bitset`.

3.24.2.10 `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc> std::basic_istream<_CharT, _Traits>& std::tr2::operator>> (std::basic_istream< _CharT, _Traits > & __is, dynamic_bitset< _WordT, _Alloc > & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace and only accept '0' and '1' characters. The `dynamic_bitset` will grow as necessary to hold the string of bits.

Definition at line 1383 of file `dynamic_bitset`.

References `std::basic_string< _CharT, _Traits, _Alloc >::empty()`, `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::tr2::dynamic_bitset< _WordT, _Alloc >::resize()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_string< _CharT, _Traits, _Alloc >::size()`, `std::tr2::dynamic_bitset< _WordT, _Alloc >::size()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

3.24.2.11 `template<typename _WordT, typename _Alloc> dynamic_bitset<_WordT, _Alloc> std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > & __x, const dynamic_bitset< _WordT, _Alloc > & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1352 of file `dynamic_bitset`.

3.24.2.12 `template<typename _WordT, typename _Alloc> dynamic_bitset<_WordT, _Alloc> std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > & __x, const dynamic_bitset< _WordT, _Alloc > & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1342 of file `dynamic_bitset`.

3.25 std::tr2::__detail Namespace Reference

3.25.1 Detailed Description

Implementation details not part of the namespace `std::tr2` interface. Dynamic Bitset.

See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library.

4 Class Documentation

4.1 __cxxabiv1::__forced_unwind Class Reference

4.1.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

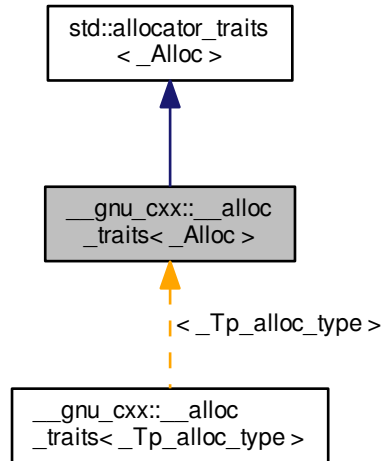
Definition at line 48 of file `cxxabi_forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi_forced.h](#)

4.2 `__gnu_cxx::__alloc_traits<_Alloc>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc>`:



Public Types

- typedef `std::allocator_traits<_Alloc>` **Base_type**
- typedef `_Alloc` **allocator_type**
- typedef `_Base_type::const_pointer` **const_pointer**
- typedef `const value_type & const_reference`
- typedef `__const_void_pointer` **const_void_pointer**
- typedef `_Base_type::difference_type` **difference_type**
- typedef `_Base_type::pointer` **pointer**
- typedef `__propagate_on_container_copy_assignment` **propagate_on_container_copy_assignment**
- typedef `__propagate_on_container_move_assignment` **propagate_on_container_move_assignment**
- typedef `__propagate_on_container_swap` **propagate_on_container_swap**
- template<typename `_Tp`>
using **rebind_alloc** = typename `__alloctr_rebind<_Alloc, _Tp>::__type`
- template<typename `_Tp`>
using **rebind_traits** = `allocator_traits<rebind_alloc<_Tp>>`
- typedef `value_type & reference`
- typedef `_Base_type::size_type` **size_type**
- typedef `_Base_type::value_type` **value_type**
- typedef `__void_pointer` **void_pointer**

Static Public Member Functions

- static constexpr bool `_S_always_equal` ()
- static constexpr bool `_S_nothrow_move` ()
- static constexpr bool `_S_nothrow_swap` ()
- static void `_S_on_swap` (_Alloc &__a, _Alloc &__b)
- static constexpr bool `_S_propagate_on_copy_assign` ()
- static constexpr bool `_S_propagate_on_move_assign` ()
- static constexpr bool `_S_propagate_on_swap` ()
- static _Alloc `_S_select_on_copy` (const _Alloc &__a)
- static [pointer allocate](#) (_Alloc &__a, [size_type](#) __n)
- static [pointer allocate](#) (_Alloc &__a, [size_type](#) __n, [const_void_pointer](#) __hint)
- template<typename _Ptr, typename... _Args>
static [std::enable_if](#)
< __is_custom_pointer< _Ptr >
::value >::type **construct** (_Alloc &__a, _Ptr __p, _Args &&... __args)
- template<typename _Tp, typename... _Args>
static auto **construct** (_Alloc &__a, _Tp *__p, _Args &&... __args) -> decltype(_S_construct(__a, __p, [std::forward](#)<_Args>(__args)...))
- static void [deallocate](#) (_Alloc &__a, [pointer](#) __p, [size_type](#) __n)
- template<typename _Ptr >
static [std::enable_if](#)
< __is_custom_pointer< _Ptr >
::value >::type **destroy** (_Alloc &__a, _Ptr __p)
- template<class _Tp >
static void [destroy](#) (_Alloc &__a, _Tp *__p)
- static [size_type max_size](#) (const _Alloc &__a)
- static _Alloc [select_on_container_copy_construction](#) (const _Alloc &__rhs)

4.2.1 Detailed Description

```
template<typename _Alloc> struct __gnu_cxx::__alloc_traits<_Alloc>
```

Uniform interface to C++98 and C++0x allocators.

Definition at line 128 of file `ext/alloc_traits.h`.

4.2.2 Member Typedef Documentation

4.2.2.1 `template<typename _Alloc> typedef __const_void_pointer std::allocator_traits<_Alloc>::const_void_pointer`
[[inherited](#)]

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 142 of file `bits/alloc_traits.h`.

4.2.2.2 `template<typename _Alloc> typedef __propagate_on_container_copy_assignment std::allocator_traits<_Alloc>::propagate_on_container_copy_assignment [inherited]`

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 176 of file `bits/alloc_traits.h`.

4.2.2.3 `template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits<_Alloc>::propagate_on_container_move_assignment [inherited]`

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 188 of file `bits/alloc_traits.h`.

4.2.2.4 `template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits<_Alloc>::propagate_on_container_swap [inherited]`

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 199 of file `bits/alloc_traits.h`.

4.2.2.5 `template<typename _Alloc> typedef __void_pointer std::allocator_traits<_Alloc>::void_pointer [inherited]`

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 131 of file `bits/alloc_traits.h`.

4.2.3 Member Function Documentation

4.2.3.1 `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate (_Alloc & __a, size_type __n) [inline], [static], [inherited]`

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 352 of file `bits/alloc_traits.h`.

4.2.3.2 `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate (_Alloc & __a, size_type __n, const_void_pointer __hint) [inline], [static], [inherited]`

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

Returns

Memory of suitable size and alignment for n objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 367 of file `bits/alloc_traits.h`.

4.2.3.3 `template<typename _Alloc> template<typename _Tp, typename... _Args> static auto std::allocator_traits<_Alloc>::construct (_Alloc & __a, _Tp * __p, _Args &&... __args) -> decltype(_S.construct(__a, __p, std::forward<_Args>(__args)...))` `[inline],[static],[inherited]`

Construct an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args) ...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 393 of file `bits/alloc_traits.h`.

4.2.3.4 `template<typename _Alloc> static void std::allocator_traits<_Alloc>::deallocate (_Alloc & __a, pointer __p, size_type __n)` `[inline],[static],[inherited]`

Deallocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the memory to deallocate.
<code>__n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 378 of file `bits/alloc_traits.h`.

4.2.3.5 `template<typename _Alloc> template<class _Tp> static void std::allocator_traits<_Alloc>::destroy (_Alloc & __a, _Tp * __p)` `[inline],[static],[inherited]`

Destroy an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 406 of file `bits/alloc_traits.h`.

4.2.3.6 `template<typename _Alloc> static size_type std::allocator_traits<_Alloc>::max_size (const _Alloc & __a)`
`[inline], [static], [inherited]`

The maximum supported allocation size.

Parameters

<code>__a</code>	An allocator.
------------------	---------------

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 417 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`.

4.2.3.7 `template<typename _Alloc> static _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction (const _Alloc & __rhs)`
`[inline], [static], [inherited]`

Obtain an allocator to use when copying a container.

Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 429 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [ext/alloc_traits.h](#)

4.3 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread>` Struct Template Reference

Inherits `__common_pool_base<_PoolTp, _Thread>`.

4.3.1 Detailed Description

`template<template< bool > class _PoolTp, bool _Thread> struct __gnu_cxx::__common_pool_policy<_PoolTp, _Thread>`

Policy for shared `__pool` objects.

Definition at line 460 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

4.4 `__gnu_cxx::__detail::__mini_vector< _Tp >` Class Template Reference

Public Types

- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator __pos) throw ()
- void **insert** (iterator __pos, const_reference __x)
- reference **operator[]** (const size_type __pos) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference __x)
- size_type **size** () const throw ()

4.4.1 Detailed Description

`template<typename _Tp>class __gnu_cxx::__detail::__mini_vector< _Tp >`

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses operator `new()` to get memory, and operator `delete()` to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

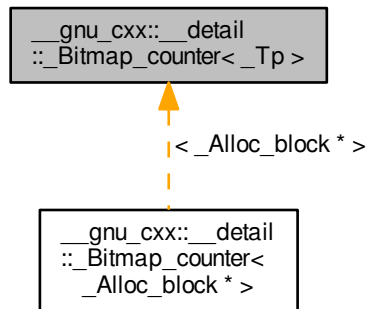
Definition at line 69 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.5 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Bitmap_counter<_Tp>`:



Public Member Functions

- `_Bitmap_counter` (`_BPVector` &Rvbp, long __index=-1)
- pointer `_M_base` () const throw ()
- bool `_M_finished` () const throw ()
- `size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- void `_M_reset` (long __index=-1) throw ()
- void `_M_set_internal_bitmap` (`size_t * __new_internal_marker`) throw ()
- `_Index_type _M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

4.5.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::__detail::_Bitmap_counter<_Tp>
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

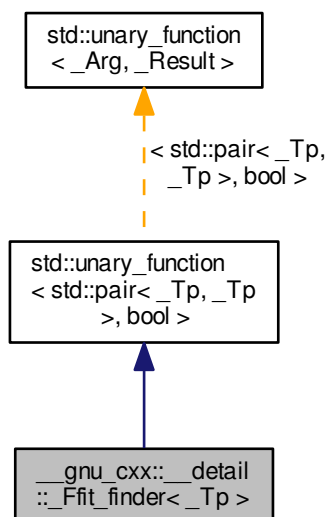
Definition at line 396 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.6 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder<_Tp>`:



Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `size_t * _M_get ()` `const throw ()`
- `_Counter_type _M_offset ()` `const throw ()`
- `bool operator() (_Block_pair __bp)` `throw ()`

4.6.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::__detail::_Ffit_finder<_Tp>
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 331 of file `bitmap_allocator.h`.

4.6.2 Member Typedef Documentation

4.6.2.1 `typedef std::pair<_Tp, _Tp> std::unary_function< std::pair<_Tp, _Tp>, bool>::argument_type`
`[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.6.2.2 `typedef bool std::unary_function< std::pair<_Tp, _Tp>, bool>::result_type` `[inherited]`

`result_type` is the return type

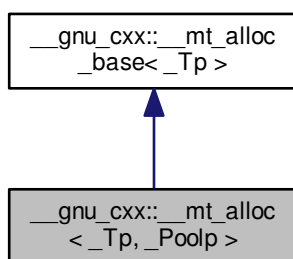
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.7 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `__mt_alloc` (const [__mt_alloc](#) &) noexcept

- `template<typename _Tp1, typename _Poolp1 >`
`__mt_alloc` (const `__mt_alloc<_Tp1, _Poolp1>` &) noexcept
- `const __pool_base::_Tune _M_get_options ()`
- `void _M_set_options (__pool_base::_Tune __t)`
- `pointer address` (reference `__x`) const noexcept
- `const_pointer address` (const_reference `__x`) const noexcept
- `pointer allocate` (size_type `__n`, const void `*=0`)
- `template<typename _Up, typename... _Args>`
`void construct` (`_Up *__p`, `_Args &&...__args`)
- `void deallocate` (pointer `__p`, size_type `__n`)
- `template<typename _Up >`
`void destroy` (`_Up *__p`)
- `size_type max_size` () const noexcept

4.7.1 Detailed Description

`template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >> class __gnu_cxx::__mt_alloc<_Tp, _Poolp>`

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

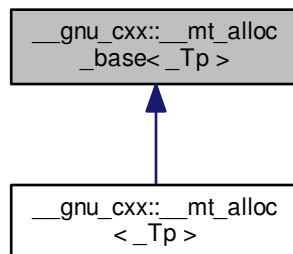
Definition at line 639 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.8 `__gnu_cxx::__mt_alloc_base<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc_base<_Tp>`:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef [std::true_type](#) **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- template<typename _Up, typename... _Args>
void **construct** (_Up *__p, _Args &&... __args)
- template<typename _Up >
void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

4.8.1 Detailed Description

template<typename _Tp>class `__gnu_cxx::__mt_alloc_base< _Tp >`

Base class for _Tp dependent member functions.

Definition at line 570 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.9 `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >` Struct Template Reference

Inherits `__per_type_pool_base< _Tp, _PoolTp, _Thread >`.

4.9.1 Detailed Description

template<typename _Tp, template< bool > class _PoolTp, bool _Thread>struct `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`

Policy for individual __pool objects.

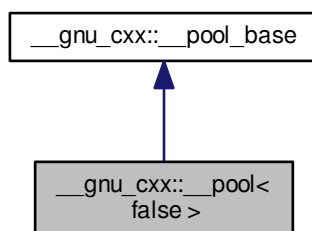
Definition at line 555 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

4.10 `__gnu_cxx::__pool< false >` Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool< false >`:



Public Types

- typedef unsigned short int **`_Binmap_type`**

Public Member Functions

- **`__pool`** (const `__pool_base::Tune &__tune`)
- void **`_M_adjust_freelist`** (const `_Bin_record &`, `_Block_record *`, `size_t`)
- bool **`_M_check_threshold`** (`size_t __bytes`)
- void **`_M_destroy`** () throw ()
- `size_t` **`_M_get_align`** ()
- const `_Bin_record &` **`_M_get_bin`** (`size_t __which`)
- `size_t` **`_M_get_binmap`** (`size_t __bytes`)
- const `_Tune &` **`_M_get_options`** () const
- `size_t` **`_M_get_thread_id`** ()
- void **`_M_initialize_once`** ()
- void **`_M_reclaim_block`** (`char *__p`, `size_t __bytes`) throw ()
- `char *` **`_M_reserve_block`** (`size_t __bytes`, const `size_t __thread_id`)
- void **`_M_set_options`** (`_Tune __t`)

Protected Attributes

- `_Binmap_type *` **`_M_binmap`**
- bool **`_M_init`**
- `_Tune` **`_M_options`**

4.10.1 Detailed Description

```
template<> class __gnu_cxx::__pool< false >
```

Specialization for single thread.

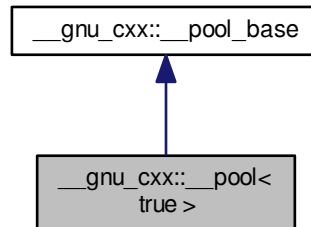
Definition at line 196 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.11 __gnu_cxx::__pool< true > Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool< true >`:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool** (const __pool_base::Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record * __block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- void **_M_destroy_thread_key** (void *) throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char * __p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

4.11.1 Detailed Description

```
template<>class __gnu_cxx::__pool< true >
```

Specialization for thread enabled, via gthreads.h.

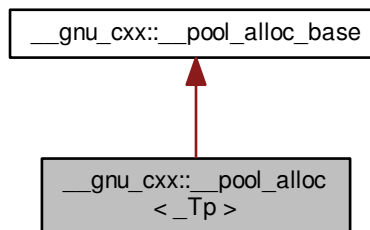
Definition at line 263 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.12 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool_alloc< _Tp >`:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `__pool_alloc` (const `__pool_alloc` &) noexcept
- `template<typename _Tp1 > __pool_alloc` (const `__pool_alloc<_Tp1>` &) noexcept
- pointer `address` (reference `__x`) const noexcept
- const_pointer `address` (const_reference `__x`) const noexcept
- pointer `allocate` (size_type `__n`, const void `*=0`)
- `template<typename _Up, typename... _Args> void construct` (`_Up *__p`, `_Args &&... __args`)
- void `deallocate` (pointer `__p`, size_type `__n`)
- `template<typename _Up > void destroy` (`_Up *__p`)
- size_type `max_size` () const noexcept

Private Types

- enum { `_S_align` }
- enum { `_S_max_bytes` }
- enum { `_S_free_list_size` }

Private Member Functions

- `char * _M_allocate_chunk` (size_t `__n`, int &`__nobjs`)
- `_Obj *volatile * _M_get_free_list` (size_t `__bytes`) throw ()
- `__mutex & _M_get_mutex` () throw ()
- `void * _M_refill` (size_t `__n`)
- `size_t _M_round_up` (size_t `__bytes`)

Static Private Attributes

- static `char * _S_end_free`
- static `_Obj *volatile _S_free_list` [`_S_free_list_size`]
- static `size_t _S_heap_size`
- static `char * _S_start_free`

4.12.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__pool_alloc<_Tp>`

Allocator using a memory pool with a single lock.

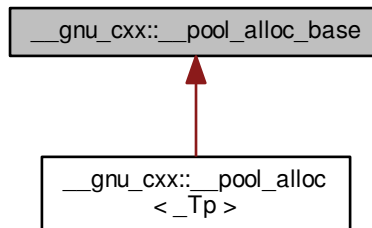
Definition at line 126 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

4.13 `__gnu_cxx::__pool_alloc_base` Class Reference

Inheritance diagram for `__gnu_cxx::__pool_alloc_base`:



Protected Types

- enum { **_S_align** }
- enum { **_S_max_bytes** }
- enum { **_S_free_list_size** }

Protected Member Functions

- char * **_M_allocate_chunk** (size_t __n, int &__nobjs)
- _Obj *volatile * **_M_get_free_list** (size_t __bytes) throw ()
- __mutex & **_M_get_mutex** () throw ()
- void * **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Protected Attributes

- static char * **_S_end_free**
- static _Obj *volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char * **_S_start_free**

4.13.1 Detailed Description

Base class for `__pool_alloc`.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size > `_S_max_bytes`, the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

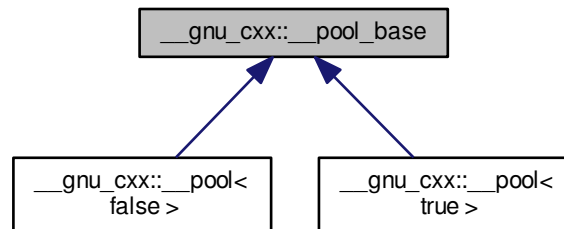
Definition at line 78 of file pool_allocator.h.

The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

4.14 __gnu_cxx::__pool_base Struct Reference

Inheritance diagram for __gnu_cxx::__pool_base:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool_base** (const _Tune &__options)
- bool **_M_check_threshold** (size_t __bytes)
- size_t **_M_get_align** ()
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

4.14.1 Detailed Description

Base class for pool object.

Definition at line 51 of file mt_allocator.h.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

4.15 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility<_CharT, _Traits, _Alloc>`.

Public Types

- typedef
 `_Util_Base::_CharT_alloc_type` **`_CharT_alloc_type`**
- typedef `__vstring_utility`
 `<_CharT, _Traits, _Alloc>` **`_Util_Base`**
- typedef `_Alloc` **`allocator_type`**
- typedef
 `_CharT_alloc_type::size_type` **`size_type`**
- typedef `_Traits` **`traits_type`**
- typedef `_Traits::char_type` **`value_type`**

Public Member Functions

- `__rc_string_base` (const `_Alloc` &`_a`)
- `__rc_string_base` (const `__rc_string_base` &`_rcs`)
- `__rc_string_base` (`__rc_string_base` &&`_rcs`)
- `__rc_string_base` (size_type `__n`, `_CharT` `__c`, const `_Alloc` &`_a`)
- template<typename `_InputIterator` >
 `__rc_string_base` (`_InputIterator` `__beg`, `_InputIterator` `__end`, const `_Alloc` &`_a`)
- void `_M_assign` (const `__rc_string_base` &`_rcs`)
- size_type `_M_capacity` () const
- void `_M_clear` ()
- bool `_M_compare` (const `__rc_string_base` &) const
- template<>
 bool `_M_compare` (const `__rc_string_base` &`_rcs`) const
- template<>
 bool `_M_compare` (const `__rc_string_base` &`_rcs`) const
- `_CharT` * `_M_data` () const
- void `_M_erase` (size_type `__pos`, size_type `__n`)
- allocator_type & `_M_get_allocator` ()
- const allocator_type & `_M_get_allocator` () const
- bool `_M_is_shared` () const
- void `_M_leak` ()
- size_type `_M_length` () const
- size_type `_M_max_size` () const
- void `_M_mutate` (size_type `__pos`, size_type `__len1`, const `_CharT` *`__s`, size_type `__len2`)
- void `_M_reserve` (size_type `__res`)
- void `_M_set_leaked` ()
- void `_M_set_length` (size_type `__n`)
- void `_M_swap` (`__rc_string_base` &`_rcs`)
- template<typename `_InIterator` >
 `_CharT` * `_S_construct` (`_InIterator` `__beg`, `_InIterator` `__end`, const `_Alloc` &`_a`, `std::forward_iterator_tag`)

Protected Types

- typedef
`__gnu_cxx::__normal_iterator`
`< const_pointer,`
`__gnu_cxx::__versa_string`
`< _CharT, _Traits, _Alloc,`
`__rc_string_base > > __const_rc_iterator`
- typedef
`__gnu_cxx::__normal_iterator`
`< const_pointer,`
`__gnu_cxx::__versa_string`
`< _CharT, _Traits, _Alloc,`
`__sso_string_base > > __const_sso_iterator`
- typedef
`__gnu_cxx::__normal_iterator`
`< pointer,`
`__gnu_cxx::__versa_string`
`< _CharT, _Traits, _Alloc,`
`__rc_string_base > > __rc_iterator`
- typedef
`__gnu_cxx::__normal_iterator`
`< pointer,`
`__gnu_cxx::__versa_string`
`< _CharT, _Traits, _Alloc,`
`__sso_string_base > > __sso_iterator`
- typedef
`_CharT_alloc_type::const_pointer` **const_pointer**
- typedef
`_CharT_alloc_type::difference_type` **difference_type**
- typedef `_CharT_alloc_type::pointer` **pointer**

Static Protected Member Functions

- static void **_S_assign** (`_CharT *__d, size_type __n, _CharT __c`)
- static int **_S_compare** (`size_type __n1, size_type __n2`)
- static void **_S_copy** (`_CharT *__d, const _CharT *__s, size_type __n`)
- template<typename `_Iterator` >
static void **_S_copy_chars** (`_CharT *__p, _Iterator __k1, _Iterator __k2`)
- static void **_S_copy_chars** (`_CharT *__p, __sso_iterator __k1, __sso_iterator __k2`)
- static void **_S_copy_chars** (`_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2`)
- static void **_S_copy_chars** (`_CharT *__p, __rc_iterator __k1, __rc_iterator __k2`)
- static void **_S_copy_chars** (`_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2`)
- static void **_S_copy_chars** (`_CharT *__p, _CharT *__k1, _CharT *__k2`)
- static void **_S_copy_chars** (`_CharT *__p, const _CharT *__k1, const _CharT *__k2`)
- static void **_S_move** (`_CharT *__d, const _CharT *__s, size_type __n`)

4.15.1 Detailed Description

template<typename `_CharT`, typename `_Traits`, typename `_Alloc`> class `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>`

Documentation? What's that? Nathan Myers ncm@cantrip.org.

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[ __rc_string_base<char_type>]      _M_capacity
_M_dataplus                        _M_refcount
_M_p ----->                      unnamed array of char_type

```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc_string_base.h](#)

4.16 `__gnu_cxx::__scoped_lock` Class Reference

Public Types

- typedef `__mutex` `__mutex_type`

Public Member Functions

- `__scoped_lock` (`__mutex_type` &`__name`)

4.16.1 Detailed Description

Scoped lock idiom.

Definition at line 231 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

4.17 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

Inherits `_Base<_CharT, _Traits, _Alloc>`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef
`__gnu_cxx::__normal_iterator`
`< const_pointer,`
`__versa_string >` **const_iterator**
- typedef
`_CharT_alloc_type::const_pointer` **const_pointer**
- typedef `const value_type &` **const_reference**
- typedef `std::reverse_iterator`
`< const_iterator >` **const_reverse_iterator**
- typedef
`_CharT_alloc_type::difference_type` **difference_type**
- typedef
`__gnu_cxx::__normal_iterator`
`< pointer, __versa_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse_iterator`
`< iterator >` **reverse_iterator**
- typedef
`_CharT_alloc_type::size_type` **size_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits::char_type` **value_type**

Public Member Functions

- `__versa_string` ()
- `__versa_string` (const `_Alloc` &__a)
- `__versa_string` (const `__versa_string` &__str)
- `__versa_string` (`__versa_string` &&__str) noexcept
- `__versa_string` (std::initializer_list< `_CharT` > __l, const `_Alloc` &__a=_Alloc())
- `__versa_string` (const `__versa_string` &__str, size_type __pos, size_type __n=npos)
- `__versa_string` (const `__versa_string` &__str, size_type __pos, size_type __n, const `_Alloc` &__a)
- `__versa_string` (const `_CharT` *__s, size_type __n, const `_Alloc` &__a=_Alloc())
- `__versa_string` (const `_CharT` *__s, const `_Alloc` &__a=_Alloc())
- `__versa_string` (size_type __n, `_CharT` __c, const `_Alloc` &__a=_Alloc())
- template<class `_InputIterator` , typename = std::_RequireInputIter< `_InputIterator` >>
`__versa_string` (`_InputIterator` __beg, `_InputIterator` __end, const `_Alloc` &__a=_Alloc())
- `~__versa_string` () noexcept
- `__versa_string` & append (const `__versa_string` &__str)
- `__versa_string` & append (const `__versa_string` &__str, size_type __pos, size_type __n)
- `__versa_string` & append (const `_CharT` *__s, size_type __n)
- `__versa_string` & append (const `_CharT` *__s)
- `__versa_string` & append (size_type __n, `_CharT` __c)
- `__versa_string` & append (std::initializer_list< `_CharT` > __l)
- template<class `_InputIterator` , typename = std::_RequireInputIter< `_InputIterator` >>
`__versa_string` & append (`_InputIterator` __first, `_InputIterator` __last)
- `__versa_string` & assign (const `__versa_string` &__str)
- `__versa_string` & assign (`__versa_string` &&__str)

- `__versa_string & assign` (const `__versa_string` & __str, size_type __pos, size_type __n)
- `__versa_string & assign` (const `_CharT *` __s, size_type __n)
- `__versa_string & assign` (const `_CharT *` __s)
- `__versa_string & assign` (size_type __n, `_CharT` __c)
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`
`__versa_string & assign` (`_InputIterator` __first, `_InputIterator` __last)
- `__versa_string & assign` (`std::initializer_list`< `_CharT` > __l)
- `const_reference at` (size_type __n) const
- `reference at` (size_type __n)
- `reference back` ()
- `const_reference back` () const
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `const _CharT * c_str` () const noexcept
- `size_type capacity` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `int compare` (const `__versa_string` & __str) const
- `int compare` (size_type __pos, size_type __n, const `__versa_string` & __str) const
- `int compare` (size_type __pos1, size_type __n1, const `__versa_string` & __str, size_type __pos2, size_type __n2) const
- `int compare` (const `_CharT *` __s) const
- `int compare` (size_type __pos, size_type __n1, const `_CharT *` __s) const
- `int compare` (size_type __pos, size_type __n1, const `_CharT *` __s, size_type __n2) const
- `size_type copy` (`_CharT *` __s, size_type __n, size_type __pos=0) const
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- `const _CharT * data` () const noexcept
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `__versa_string & erase` (size_type __pos=0, size_type __n=`npos`)
- `iterator erase` (iterator __position)
- `iterator erase` (iterator __first, iterator __last)
- `size_type find` (const `_CharT *` __s, size_type __pos, size_type __n) const
- `size_type find` (const `__versa_string` & __str, size_type __pos=0) const noexcept
- `size_type find` (const `_CharT *` __s, size_type __pos=0) const
- `size_type find` (`_CharT` __c, size_type __pos=0) const noexcept
- `size_type find_first_not_of` (const `__versa_string` & __str, size_type __pos=0) const noexcept
- `size_type find_first_not_of` (const `_CharT *` __s, size_type __pos, size_type __n) const
- `size_type find_first_not_of` (const `_CharT *` __s, size_type __pos=0) const
- `size_type find_first_not_of` (`_CharT` __c, size_type __pos=0) const noexcept
- `size_type find_first_of` (const `__versa_string` & __str, size_type __pos=0) const noexcept
- `size_type find_first_of` (const `_CharT *` __s, size_type __pos, size_type __n) const
- `size_type find_first_of` (const `_CharT *` __s, size_type __pos=0) const
- `size_type find_first_of` (`_CharT` __c, size_type __pos=0) const noexcept
- `size_type find_last_not_of` (const `__versa_string` & __str, size_type __pos=`npos`) const noexcept
- `size_type find_last_not_of` (const `_CharT *` __s, size_type __pos, size_type __n) const
- `size_type find_last_not_of` (const `_CharT *` __s, size_type __pos=`npos`) const
- `size_type find_last_not_of` (`_CharT` __c, size_type __pos=`npos`) const noexcept

- `size_type find_last_of` (const `__versa_string` & `__str`, `size_type __pos=npos`) const noexcept
- `size_type find_last_of` (const `_CharT *` `__s`, `size_type __pos`, `size_type __n`) const
- `size_type find_last_of` (const `_CharT *` `__s`, `size_type __pos=npos`) const
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npos`) const noexcept
- reference `front` ()
- const_reference `front` () const
- allocator_type `get_allocator` () const noexcept
- void `insert` (iterator `__p`, `size_type __n`, `_CharT __c`)
- template<class `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`
void `insert` (iterator `__p`, `_InputIterator __beg`, `_InputIterator __end`)
- void `insert` (iterator `__p`, `std::initializer_list<_CharT> __l`)
- `__versa_string` & `insert` (`size_type __pos1`, const `__versa_string` & `__str`)
- `__versa_string` & `insert` (`size_type __pos1`, const `__versa_string` & `__str`, `size_type __pos2`, `size_type __n`)
- `__versa_string` & `insert` (`size_type __pos`, const `_CharT *` `__s`, `size_type __n`)
- `__versa_string` & `insert` (`size_type __pos`, const `_CharT *` `__s`)
- `__versa_string` & `insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- iterator `insert` (iterator `__p`, `_CharT __c`)
- `size_type length` () const noexcept
- `size_type max_size` () const noexcept
- `__versa_string` & `operator+=` (const `__versa_string` & `__str`)
- `__versa_string` & `operator+=` (const `_CharT *` `__s`)
- `__versa_string` & `operator+=` (`_CharT __c`)
- `__versa_string` & `operator+=` (`std::initializer_list<_CharT> __l`)
- `__versa_string` & `operator=` (const `__versa_string` & `__str`)
- `__versa_string` & `operator=` (`__versa_string` && `__str`)
- `__versa_string` & `operator=` (`std::initializer_list<_CharT> __l`)
- `__versa_string` & `operator=` (const `_CharT *` `__s`)
- `__versa_string` & `operator=` (`_CharT __c`)
- const_reference `operator[]` (`size_type __pos`) const
- reference `operator[]` (`size_type __pos`)
- void `pop_back` ()
- void `push_back` (`_CharT __c`)
- `reverse_iterator rbegin` () noexcept
- `const_reverse_iterator rbegin` () const noexcept
- `reverse_iterator rend` () noexcept
- `const_reverse_iterator rend` () const noexcept
- `__versa_string` & `replace` (`size_type __pos`, `size_type __n`, const `__versa_string` & `__str`)
- `__versa_string` & `replace` (`size_type __pos1`, `size_type __n1`, const `__versa_string` & `__str`, `size_type __pos2`, `size_type __n2`)
- `__versa_string` & `replace` (`size_type __pos`, `size_type __n1`, const `_CharT *` `__s`, `size_type __n2`)
- `__versa_string` & `replace` (`size_type __pos`, `size_type __n1`, const `_CharT *` `__s`)
- `__versa_string` & `replace` (`size_type __pos`, `size_type __n1`, `size_type __n2`, `_CharT __c`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, const `__versa_string` & `__str`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, const `_CharT *` `__s`, `size_type __n`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, const `_CharT *` `__s`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, `size_type __n`, `_CharT __c`)
- template<class `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`
`__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, `_InputIterator __k1`, `_InputIterator __k2`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, `_CharT *` `__k1`, `_CharT *` `__k2`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, const `_CharT *` `__k1`, const `_CharT *` `__k2`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, iterator `__k1`, iterator `__k2`)

- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, const_iterator `__k1`, const_iterator `__k2`)
- `__versa_string` & `replace` (iterator `__i1`, iterator `__i2`, `std::initializer_list<_CharT>` `__l`)
- void `reserve` (size_type `__res_arg=0`)
- void `resize` (size_type `__n`, `_CharT` `__c`)
- void `resize` (size_type `__n`)
- size_type `rfind` (const `__versa_string` & `__str`, size_type `__pos=npos`) const noexcept
- size_type `rfind` (const `_CharT` * `__s`, size_type `__pos`, size_type `__n`) const
- size_type `rfind` (const `_CharT` * `__s`, size_type `__pos=npos`) const
- size_type `rfind` (`_CharT` `__c`, size_type `__pos=npos`) const noexcept
- void `shrink_to_fit` ()
- size_type `size` () const noexcept
- `__versa_string` `substr` (size_type `__pos=0`, size_type `__n=npos`) const
- void `swap` (`__versa_string` & `__s`)

Static Public Attributes

- static const size_type `npos`

4.17.1 Detailed Description

template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`>class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

Definition at line 56 of file `vstring.h`.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`> `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ()` `[inline]`

Default constructor creates an empty string.

Definition at line 134 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr()`.

4.17.2.2 template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`> `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (const _Alloc & __a)` `[inline]`, `[explicit]`

Construct an empty string using allocator `a`.

Definition at line 141 of file `vstring.h`.

4.17.2.3 template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`> `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str)` `[inline]`

Construct string with copy of value of `__str`.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 149 of file `vstring.h`.

4.17.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string(__versa_string<_CharT, _Traits, _Alloc, _Base> && __str) [inline], [noexcept]`

String move constructor.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

Definition at line 161 of file `vstring.h`.

4.17.2.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string(std::initializer_list<_CharT> __l, const _Alloc & __a = _Alloc()) [inline]`

Construct string from an initializer list.

Parameters

<code>__l</code>	<code>std::initializer_list</code> of characters.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 169 of file `vstring.h`.

4.17.2.6 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string(const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos, size_type __n = npos) [inline]`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy (default remainder).

Definition at line 180 of file `vstring.h`.

4.17.2.7 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string(const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos, size_type __n, const _Alloc & __a) [inline]`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.

Definition at line 195 of file `vstring.h`.

4.17.2.8 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (const _CharT * __s, size_type __n, const _Alloc & __a = _Alloc()) [inline]`

Construct string initialized by a character array.

Parameters

<code>__s</code>	Source character array.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 212 of file `vstring.h`.

4.17.2.9 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (const _CharT * __s, const _Alloc & __a = _Alloc()) [inline]`

Construct string as copy of a C string.

Parameters

<code>__s</code>	Source C string.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 221 of file `vstring.h`.

4.17.2.10 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (size_type __n, _CharT __c, const _Alloc & __a = _Alloc()) [inline]`

Construct string as multiple characters.

Parameters

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 231 of file `vstring.h`.

4.17.2.11 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (_InputIterator __beg, _InputIterator __end, const _Alloc & __a = _Alloc()) [inline]`

Construct string as copy of a range.

Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 246 of file `vstring.h`.

4.17.2.12 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~__versa_string () [inline], [noexcept]`

Destroy the string instance.

Definition at line 253 of file `vstring.h`.

4.17.3 Member Function Documentation

4.17.3.1 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 688 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `std::getline()`, `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=()`.

4.17.3.2 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos, size_type __n) [inline]`

Append a substring.

Parameters

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <i>pos</i> is not a valid index.
--------------------------------	-------------------------------------

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 705 of file `vstring.h`.

4.17.3.3 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const _CharT * __s, size_type __n) [inline]`

Append a C substring.

Parameters

<code>__s</code>	The C string to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

Definition at line 717 of file `vstring.h`.

4.17.3.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const _CharT * __s) [inline]`

Append a C string.

Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

Returns

Reference to this string.

Definition at line 730 of file `vstring.h`.

4.17.3.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (size_type __n, _CharT __c) [inline]`

Append multiple characters.

Parameters

<code>__n</code>	The number of characters to append.
<code>__c</code>	The character to use.

Returns

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 747 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.17.3.6 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (std::initializer_list<_CharT> __l) [inline]`

Append an `initializer_list` of characters.

Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to append.
------------------	--

Returns

Reference to this string.

Definition at line 757 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

4.17.3.7 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (_InputIterator __first, _InputIterator __last) [inline]`

Append a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range `[first,last)` to this string.

Definition at line 776 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.8 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 799 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator=()`.

4.17.3.9 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (__versa_string<_CharT, _Traits, _Alloc, _Base > && __str) [inline]`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 815 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

4.17.3.10 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos, size_type __n) [inline]`

Set value to a substring of a string.

Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of <code>str</code> .
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 836 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.11 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (const _CharT * __s, size_type __n) [inline]`

Set value to a C substring.

Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 853 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.12 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (const _CharT * __s) [inline]`

Set value to contents of a C string.

Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 869 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.13 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (size_type __n, _CharT __c) [inline]`

Set value to multiple characters.

Parameters

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 886 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.17.3.14 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (_InputIterator __first, _InputIterator __last) [inline]`

Set value to a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 905 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.15 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (std::initializer_list<_CharT> __l) [inline]`

Set value to an `initializer_list` of characters.

Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to assign.
------------------	--

Returns

Reference to this string.

Definition at line 915 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

4.17.3.16 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at (size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

Returns

Read-only (const) reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 579 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.17 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

Returns

Read/write reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 598 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.18 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::back () [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 628 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.19 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::back () const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 636 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.20 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () [inline], [noexcept]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 319 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend()`.

4.17.3.21 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 330 of file `vstring.h`.

4.17.3.22 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str () const [inline], [noexcept]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1538 of file `vstring.h`.

4.17.3.23 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity () const [inline], [noexcept]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 490 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit()`.

4.17.3.24 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 394 of file `vstring.h`.

4.17.3.25 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 402 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.26 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::clear () [inline], [noexcept]`

Erases the string, making it empty.

Definition at line 518 of file `vstring.h`.

4.17.3.27 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) const [inline]`

Compare to a string.

Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1964 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

4.17.3.28 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (size_type __pos, size_type __n, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) const`

Compare substring to a string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 459 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.3.29 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of <code>str</code> .
<code>__n2</code>	Number of characters in substring of <code>str</code> .

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(), str.substr(pos2, n2).data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 476 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `std::min()`.

4.17.3.30 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (const _CharT * __s) const`

Compare to a C string.

Parameters

<code>__s</code>	C string to compare against.
------------------	------------------------------

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(), s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 495 of file `vstring.tcc`.

References `std::min()`, and `std::size()`.

4.17.331 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (size_type __pos, size_type __n1, const _CharT * __s) const`

Compare substring to a C string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 511 of file `vstring.tcc`.

References `std::min()`.

4.17.332 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) const`

Compare substring against a character array.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of <code>s</code> .

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `\0` has no special meaning.

Definition at line 528 of file `vstring.tcc`.

References `std::min()`.

4.17.33 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::copy (_CharT * __s, size_type __n, size_type __pos = 0) const`

Copy substring into C string.

Parameters

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

<code>std::out_of_range</code>	If <code>pos > size()</code> .
--------------------------------	-----------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 254 of file `vstring.tcc`.

4.17.34 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::crbegin () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 411 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

4.17.35 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::crend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 420 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`.

4.17.36 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data () const [inline], [noexcept]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1548 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

4.17.337 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::empty () const [inline], [noexcept]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 526 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.338 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::end () [inline], [noexcept]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 338 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::crbegin()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rbegin()`.

4.17.339 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::end () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 349 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.17.340 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::erase (size_type __pos = 0, size_type __n = npos) [inline]`

Remove characters.

Parameters

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
--------------------------------	---

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1115 of file `vstring.h`.

Referenced by `std::getline()`.

4.17.3.41 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (iterator __position) [inline]`

Remove one character.

Parameters

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1131 of file `vstring.h`.

4.17.3.42 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (iterator __first, iterator __last) [inline]`

Remove a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1152 of file `vstring.h`.

4.17.3.43 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a C substring.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>__s</code> to search for.

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 269 of file `vstring.tcc`.

References `std::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

4.17.3.44 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0) const [inline], [noexcept]`

Find position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1584 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

4.17.3.45 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1599 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

4.17.3.46 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find (_CharT __c, size_type __pos = 0) const [noexcept]`

Find position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 293 of file `vstring.tcc`.

References `std::size()`.

```
4.17.3.47  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of ( const
           __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const    [inline], [noexcept]
```

Find position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1816 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`.

```
4.17.3.48  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> __versa_string<_CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string<_CharT, _Traits,
           _Alloc, _Base >::find_first_not_of ( const _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character not in C substring.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 391 of file `vstring.tcc`.

References `std::size()`.

```
4.17.3.49 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
        _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of ( const _CharT *
        __s, size_type __pos = 0 ) const    [inline]
```

Find position of a character not in C string.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1847 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

```
4.17.3.50 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
        _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits,
        _Alloc, _Base >::find_first_not_of ( _CharT __c, size_type __pos = 0 ) const    [noexcept]
```

Find position of a different character.

Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 404 of file `vstring.tcc`.

References `std::size()`.

```
4.17.3.51 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of ( const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const    [inline], [noexcept]
```

Find position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1689 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

```
4.17.3.52 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::find_first_of ( const _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character of C substring.

Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 352 of file `vstring.tcc`.

References `std::size()`.

4.17.353 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1719 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

4.17.354 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of (_CharT __c, size_type __pos = 0) const [inline], [noexcept]`

Find position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1738 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

4.17.355 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos) const [inline], [noexcept]`

Find last position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1879 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

4.17.3.56 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character not in C substring.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 416 of file `vstring.tcc`.

References `std::size()`.

4.17.3.57 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a character not in C string.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1910 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

4.17.3.58 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of (_CharT __c, size_type __pos = npos) const [noexcept]`

Find last position of a different character.

Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 438 of file `vstring.tcc`.

References `std::size()`.

4.17.3.59 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = npos) const [inline], [noexcept]`

Find last position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1753 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

4.17.3.60 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character of C substring.

Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 369 of file `vstring.tcc`.

References `std::size()`.

```
4.17.3.61 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of ( const _CharT * __s,
size_type __pos = npos ) const [inline]
```

Find last position of a character of C string.

Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1783 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

```
4.17.3.62 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of ( _CharT __c, size_type
__pos = npos ) const [inline], [noexcept]
```

Find last position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1802 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

4.17.3.63 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::front () [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 612 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`.

4.17.3.64 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::front () const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 620 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`.

4.17.3.65 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> allocator_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::get_allocator () const [inline], [noexcept]`

Return copy of allocator used to construct this string.

Definition at line 1555 of file `vstring.h`.

4.17.3.66 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (iterator __p, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 933 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.67 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (iterator __p, _InputIterator __beg, _InputIterator __end) [inline]`

Insert a range of characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 955 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.68 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (iterator __p, std::initializer_list<_CharT> __l) [inline]`

Insert an `initializer_list` of characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 966 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

4.17.3.69 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Insert value of a string.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 983 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.17.3.70 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n) [inline]`

Insert a substring.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in <code>str</code> to insert.
<code>__n</code>	Number of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos1 > size()</code> or <code>__pos2 > __str.size()</code> .

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1006 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.71 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, const _CharT* __s, size_type __n) [inline]`

Insert a C substring.

Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.
<code>__n</code>	The number of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1029 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.72 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, const _CharT* __s) [inline]`

Insert a C string.

Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1048 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.17.3.73 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1072 of file `vstring.h`.

4.17.3.74 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (iterator __p, _CharT __c) [inline]`

Insert one character.

Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1090 of file `vstring.h`.

4.17.3.75 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::length () const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 435 of file `vstring.h`.

4.17.3.76 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::max_size () const [inline], [noexcept]`

Returns the `size()` of the largest possible string.

Definition at line 440 of file `vstring.h`.

Referenced by `std::getline()`.

4.17.3.77 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 647 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

4.17.3.78 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(const _CharT * __s) [inline]`

Append a C string.

Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

Returns

Reference to this string.

Definition at line 656 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

4.17.3.79 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(_CharT __c) [inline]`

Append a character.

Parameters

<code>__c</code>	The character to append.
------------------	--------------------------

Returns

Reference to this string.

Definition at line 665 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`.

```
4.17.3.80 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+=(
std::initializer_list<_CharT> &_I ) [inline]
```

Append an `initializer_list` of characters.

Parameters

<code>__I</code>	The <code>initializer_list</code> of characters to be appended.
------------------	---

Returns

Reference to this string.

Definition at line 678 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

```
4.17.3.81 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator= ( const
__versa_string<_CharT, _Traits, _Alloc, _Base> &_str ) [inline]
```

Assign the value of `str` to this string.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 260 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

```
4.17.3.82 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator= (
__versa_string<_CharT, _Traits, _Alloc, _Base> &&_str ) [inline]
```

String move assignment operator.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 272 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

4.17.3.83 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (std::initializer_list< _CharT > _I) [inline]`

Set value to string constructed from initializer list.

Parameters

<code>__I</code>	std::initializer_list.
------------------	------------------------

Definition at line 284 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

4.17.3.84 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (const _CharT *_s) [inline]`

Copy contents of `__s` into this string.

Parameters

<code>__s</code>	Source null-terminated string.
------------------	--------------------------------

Definition at line 296 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

4.17.3.85 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (_CharT __c) [inline]`

Set value to string of length 1.

Parameters

<code>__c</code>	Source character.
------------------	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 307 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

4.17.3.86 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (size_type __pos) const [inline]`

Subscript access to the data contained in the string.

Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 541 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::front()`.

4.17.3.87 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[] (size_type __pos) [inline]`

Subscript access to the data contained in the string.

Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 558 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.17.3.88 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::pop_back () [inline]`

Remove the last character.

The string must be non-empty.

Definition at line 1169 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.17.3.89 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back (_CharT __c) [inline]`

Append a single character.

Parameters

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 784 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(())`.

4.17.3.90 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rbegin ()`
`[inline], [noexcept]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 358 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::end()`.

4.17.3.91 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rbegin () const`
`[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 367 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::end()`.

4.17.3.92 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rend ()` `[inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 376 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

4.17.3.93 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rend () const`
`[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 385 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

4.17.3.94 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str)` `[inline]`

Replace characters with value from another string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1191 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.17.3.95 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n2) [inline]`

Replace characters with value from another string.

Parameters

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of <code>str</code> to use.
<code>__n2</code>	Number of characters from <code>str</code> to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos1 > size()</code> or <code>__pos2 > str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos1,pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1214 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.17.3.96 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) [inline]`

Replace characters with value of a C substring.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.
<code>__n2</code>	Number of characters from <code>__s</code> to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos1 > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1242 of file `vstring.h`.

4.17.3.97 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n1, const _CharT * __s) [inline]`

Replace characters with value of a C string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1266 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.17.3.98 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c) [inline]`

Replace characters with multiple characters.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1290 of file `vstring.h`.

4.17.3.99 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Replace range of characters with string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1, i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1308 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.17.3.100 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n) [inline]`

Replace range of characters with C substring.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1326 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.17.3.101 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, const _CharT * __s) [inline]`

Replace range of characters with C string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1347 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.17.3.102 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, size_type __n, _CharT __c) [inline]`

Replace range of characters with multiple characters.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1368 of file `vstring.h`.

4.17.3.103 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2) [inline]`

Replace range of characters with range.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1393 of file `vstring.h`.

```
4.17.3.104 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( iterator __i1,
iterator __i2, std::initializer_list<_CharT> __l ) [inline]
```

Replace range of characters with `initializer_list`.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1474 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

```
4.17.3.105 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve ( size_type __res_arg = 0 )
[inline]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

Exceptions

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 511 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

4.17.3.106 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize (size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 50 of file `vstring.tcc`.

References `std::size()`.

4.17.3.107 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize (size_type __n) [inline]`

Resizes the string to the specified number of characters.

Parameters

<code>__n</code>	Number of characters the string should contain.
------------------	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 467 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

4.17.3.108 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = npos) const [inline], [noexcept]`

Find last position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1629 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

4.17.3.109 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a C substring.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 312 of file `vstring.tcc`.

References `std::min()`, and `std::size()`.

4.17.3.110 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a C string.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1659 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

4.17.3.111 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (_CharT __c, size_type __pos = npos) const [noexcept]`

Find last position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 334 of file `vstring.tcc`.

References `std::size()`.

4.17.3.112 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit () [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 473 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.17.3.113 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size () const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 429 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::cend()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::empty()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::pop_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

4.17.3.114 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr (size_type __pos = 0, size_type __n = npos) const [inline]`

Get a substring.

Parameters

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

<code>std::out_of_range</code>	If <code>pos > size()</code> .
--------------------------------	-----------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 1943 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string()`.

4.17.3.115 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & __s) [inline]`

Swap contents with another string.

Parameters

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1527 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`, and `__gnu_cxx::swap()`.

4.17.4 Member Data Documentation

4.17.4.1 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 81 of file `vstring.h`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

4.18 `__gnu_cxx::Caster<_ToType>` Struct Template Reference

Public Types

- `typedef _ToType::element_type * type`

4.18.1 Detailed Description

`template<typename _ToType>struct __gnu_cxx::Caster<_ToType>`

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names

the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file `cast.h`.

The documentation for this struct was generated from the following file:

- [cast.h](#)

4.19 `__gnu_cxx::Char_types<_CharT>` Struct Template Reference

Public Types

- typedef unsigned long **int_type**
- typedef [std::streamoff](#) **off_type**
- typedef [std::streampos](#) **pos_type**
- typedef `std::mbstate_t` **state_type**

4.19.1 Detailed Description

```
template<typename _CharT> struct __gnu_cxx::Char_types<_CharT>
```

Mapping from character type to associated types.

Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, streamoff, streampos, and mbstate_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 58 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.20 `__gnu_cxx::ExtPtr_allocator<_Tp>` Class Template Reference

Public Types

- typedef [_Pointer_adapter](#)
 < [_Relative_pointer_impl](#)
 < const _Tp > > **const_pointer**
- typedef const _Tp & **const_reference**
- typedef `std::ptrdiff_t` **difference_type**
- typedef [_Pointer_adapter](#)
 < [_Relative_pointer_impl](#) < _Tp > > **pointer**
- typedef _Tp & **reference**
- typedef `std::size_t` **size_type**
- typedef _Tp **value_type**

Public Member Functions

- `_ExtPtr_allocator` (const `_ExtPtr_allocator` &__rarg) noexcept
- `template<typename _Up >`
`_ExtPtr_allocator` (const `_ExtPtr_allocator`< `_Up` > &__rarg) noexcept
- `const std::allocator< _Tp > & _M_getUnderlyingImp ()` const
- `pointer address` (reference __x) const noexcept
- `const_pointer address` (const_reference __x) const noexcept
- `pointer allocate` (size_type __n, void * __hint=0)
- `template<typename _Up, typename... _Args>`
`void construct` (`_Up` * __p, `_Args` &&... __args)
- `template<typename... _Args>`
`void construct` (`pointer` __p, `_Args` &&... __args)
- `void deallocate` (`pointer` __p, size_type __n)
- `template<typename _Up >`
`void destroy` (`_Up` * __p)
- `void destroy` (`pointer` __p)
- `size_type max_size` () const noexcept
- `template<typename _Up >`
`bool operator!=` (const `_ExtPtr_allocator`< `_Up` > &__rarg)
- `bool operator!=` (const `_ExtPtr_allocator` &__rarg)
- `template<typename _Up >`
`bool operator==` (const `_ExtPtr_allocator`< `_Up` > &__rarg)
- `bool operator==` (const `_ExtPtr_allocator` &__rarg)

Friends

- `template<typename _Up >`
`void swap` (`_ExtPtr_allocator`< `_Up` > &, `_ExtPtr_allocator`< `_Up` > &)

4.20.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::ExtPtr_allocator< _Tp >
```

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.

Definition at line 56 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr_allocator.h](#)

4.21 `__gnu_cxx::Invalid_type` Struct Reference

4.21.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

Definition at line 213 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

4.22 `__gnu_cxx::Pointer_adapter<_Storage_policy>` Class Template Reference

Inherits `_Storage_policy`.

Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef `_Storage_policy::element_type` **element_type**
- typedef `std::random_access_iterator_tag` **iterator_category**
- typedef `_Pointer_adapter` **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef `_Unqualified_type< element_type >::type` **value_type**

Public Member Functions

- **_Pointer_adapter** (`element_type * __arg=0`)
- **_Pointer_adapter** (`const _Pointer_adapter & __arg`)
- `template<typename _Up >`
_Pointer_adapter (`_Up * __arg`)
- `template<typename _Up >`
_Pointer_adapter (`const _Pointer_adapter<_Up> & __arg`)
- **operator __unspecified_bool_type** () const
- **bool operator!** () const
- **reference operator*** () const
- `_Pointer_adapter & operator++` ()
- `_Pointer_adapter operator++` (int)
- `_Pointer_adapter & operator+=` (short `__offset`)
- `_Pointer_adapter & operator+=` (unsigned short `__offset`)
- `_Pointer_adapter & operator+=` (int `__offset`)
- `_Pointer_adapter & operator+=` (unsigned int `__offset`)
- `_Pointer_adapter & operator+=` (long `__offset`)
- `_Pointer_adapter & operator+=` (unsigned long `__offset`)
- `template<typename _Up >`
`std::ptrdiff_t operator-` (`const _Pointer_adapter<_Up> & __rhs`) const
- `_Pointer_adapter & operator--` ()
- `_Pointer_adapter operator--` (int)
- `_Pointer_adapter & operator-=` (short `__offset`)
- `_Pointer_adapter & operator-=` (unsigned short `__offset`)
- `_Pointer_adapter & operator-=` (int `__offset`)
- `_Pointer_adapter & operator-=` (unsigned int `__offset`)
- `_Pointer_adapter & operator-=` (long `__offset`)
- `_Pointer_adapter & operator-=` (unsigned long `__offset`)

- `element_type * operator-> () const`
- `_Pointer_adapter & operator= (const _Pointer_adapter &__arg)`
- `template<typename _Up >
_Pointer_adapter & operator= (const _Pointer_adapter<_Up> &__arg)`
- `template<typename _Up >
_Pointer_adapter & operator= (_Up *__arg)`
- `reference operator[] (std::ptrdiff_t __index) const`

Friends

- `_Pointer_adapter operator+ (const _Pointer_adapter &__lhs, short __offset)`
- `_Pointer_adapter operator+ (short __offset, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator+ (const _Pointer_adapter &__lhs, unsigned short __offset)`
- `_Pointer_adapter operator+ (unsigned short __offset, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator+ (const _Pointer_adapter &__lhs, int __offset)`
- `_Pointer_adapter operator+ (int __offset, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator+ (const _Pointer_adapter &__lhs, unsigned int __offset)`
- `_Pointer_adapter operator+ (unsigned int __offset, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator+ (const _Pointer_adapter &__lhs, long __offset)`
- `_Pointer_adapter operator+ (long __offset, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator+ (unsigned long __offset, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator+ (const _Pointer_adapter &__lhs, unsigned long __offset)`
- `std::ptrdiff_t operator- (const _Pointer_adapter &__lhs, element_type *__rhs)`
- `std::ptrdiff_t operator- (element_type *__lhs, const _Pointer_adapter &__rhs)`
- `template<typename _Up >
std::ptrdiff_t operator- (const _Pointer_adapter &__lhs, _Up *__rhs)`
- `template<typename _Up >
std::ptrdiff_t operator- (_Up *__lhs, const _Pointer_adapter &__rhs)`
- `_Pointer_adapter operator- (const _Pointer_adapter &__lhs, short __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter &__lhs, unsigned short __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter &__lhs, int __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter &__lhs, unsigned int __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter &__lhs, long __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter &__lhs, unsigned long __offset)`

4.22.1 Detailed Description

```
template<typename _Storage_policy> class __gnu_cxx::Pointer_adapter<_Storage_policy>
```

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp> >; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const _Pointer_
adapter<_Std_pointer_impl<const _Tp> >;
```

Definition at line 281 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.23 `__gnu_cxx::Relative_pointer_impl<_Tp>` Class Template Reference

Public Types

- typedef `_Tp` **element_type**

Public Member Functions

- `_Tp * get () const`
- `bool operator< (const _Relative_pointer_impl &__rarg) const`
- `bool operator== (const _Relative_pointer_impl &__rarg) const`
- `void set (_Tp *__arg)`

4.23.1 Detailed Description

```
template<typename _Tp> class __gnu_cxx::Relative_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 109 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.24 `__gnu_cxx::Relative_pointer_impl<const _Tp>` Class Template Reference

Public Types

- typedef `const _Tp` **element_type**

Public Member Functions

- `const _Tp * get () const`
- `bool operator< (const _Relative_pointer_impl &__rarg) const`

- bool **operator==** (const [_Relative_pointer_impl](#) &__rarg) const
- void **set** (const _Tp *__arg)

4.24.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::Relative_pointer_impl< const _Tp >
```

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 161 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.25 `__gnu_cxx::Std_pointer_impl<_Tp>` Class Template Reference

Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp * **get** () const
- bool **operator<** (const [_Std_pointer_impl](#) &__rarg) const
- bool **operator==** (const [_Std_pointer_impl](#) &__rarg) const
- void **set** (element_type *__arg)

4.25.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::Std_pointer_impl< _Tp >
```

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

Definition at line 66 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.26 `__gnu_cxx::Unqualified_type<_Tp>` Struct Template Reference

Public Types

- typedef _Tp **type**

4.26.1 Detailed Description

```
template<typename _Tp>struct __gnu_cxx::Unqualified_type< _Tp >
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

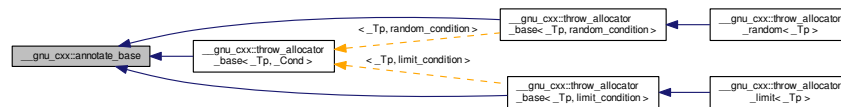
Definition at line 241 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

4.27 __gnu_cxx::annotate_base Struct Reference

Inheritance diagram for `__gnu_cxx::annotate_base`:



Public Member Functions

- void **check_allocated** (void *p, size_t size)
- void **check_allocated** (size_t label)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- [std::ostream](#) & **operator**<< ([std::ostream](#) &, const [annotate_base](#) &)

4.27.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

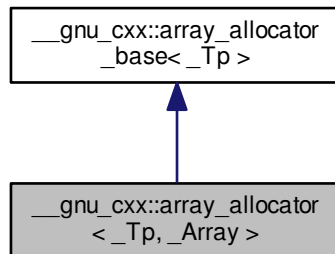
Definition at line 88 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.28 `__gnu_cxx::array_allocator< _Tp, _Array >` Class Template Reference

Inheritance diagram for `__gnu_cxx::array_allocator< _Tp, _Array >`:



Public Types

- typedef `_Array` **array_type**
- typedef const `_Tp *` **const_pointer**
- typedef const `_Tp &` **const_reference**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Tp *` **pointer**
- typedef `std::true_type` **propagate_on_container_move_assignment**
- typedef `_Tp &` **reference**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **array_allocator** (`array_type *__array=0`) noexcept
- **array_allocator** (const `array_allocator` &__o) noexcept
- template<typename `_Tp1` , typename `_Array1` >
array_allocator (const `array_allocator`< `_Tp1`, `_Array1` > &) noexcept
- pointer **address** (reference `__x`) const noexcept
- const_pointer **address** (const_reference `__x`) const noexcept
- pointer **allocate** (`size_type __n`, const void *=0)
- template<typename `_Up` , typename... `_Args`>
void **construct** (`_Up *__p`, `_Args` &&...__args)
- void **deallocate** (pointer, `size_type`)
- template<typename `_Up` >
void **destroy** (`_Up *__p`)
- `size_type` **max_size** () const noexcept

4.28.1 Detailed Description

```
template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>> class __gnu_cxx::array_allocator<_Tp, _Array>
```

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

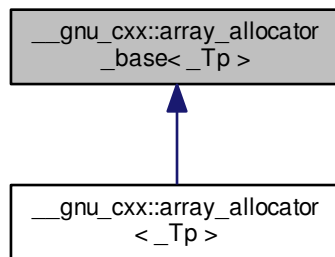
Definition at line 106 of file `array_allocator.h`.

The documentation for this class was generated from the following file:

- [array_allocator.h](#)

4.29 `__gnu_cxx::array_allocator_base<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::array_allocator_base<_Tp>`:



Public Types

- typedef const `_Tp` * **const_pointer**
- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef `_Tp` * **pointer**
- typedef `_Tp` & **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- pointer **address** (reference `__x`) const noexcept
- const_pointer **address** (const_reference `__x`) const noexcept
- template<typename `_Up`, typename... `_Args`>
void **construct** (`_Up` *`__p`, `_Args` &&...`__args`)
- void **deallocate** (pointer, size_type)
- template<typename `_Up`>
void **destroy** (`_Up` *`__p`)
- size_type **max_size** () const noexcept

4.29.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::array_allocator_base< _Tp >
```

Base class.

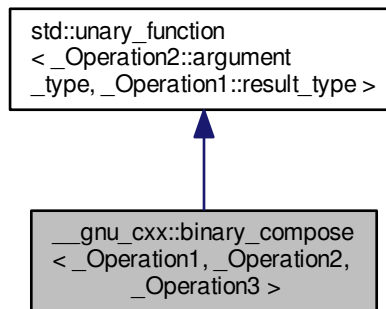
Definition at line 50 of file `array_allocator.h`.

The documentation for this class was generated from the following file:

- [array_allocator.h](#)

4.30 `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >` Class Template Reference

Inheritance diagram for `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **binary_compose** (const `_Operation1` &__x, const `_Operation2` &__y, const `_Operation3` &__z)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &__x) const

Protected Attributes

- `_Operation1` **_M_fn1**
- `_Operation2` **_M_fn2**
- `_Operation3` **_M_fn3**

4.30.1 Detailed Description

```
template<class _Operation1, class _Operation2, class _Operation3>class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
```

An [SGI extension](#) .

Definition at line 150 of file ext/functional.

4.30.2 Member Typedef Documentation

4.30.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`
[inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.30.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`
[inherited]

`result_type` is the return type

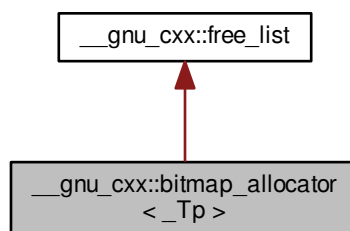
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- [ext/functional](#)

4.31 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference

Inheritance diagram for `__gnu_cxx::bitmap_allocator< _Tp >`:



Public Types

- `typedef free_list::__mutex_type __mutex_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`

- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef [std::true_type](#) **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **bitmap_allocator** (const [bitmap_allocator](#) &) noexcept
- template<typename _Tp1 >
 bitmap_allocator (const [bitmap_allocator](#)<_Tp1> &) noexcept
- pointer [_M_allocate_single_object](#) () throw (std::bad_alloc)
- void [_M_deallocate_single_object](#) (pointer __p) throw ()
- pointer **address** (reference __r) const noexcept
- const_pointer **address** (const_reference __r) const noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, typename [bitmap_allocator](#)< void >::const_pointer)
- template<typename _Up, typename... _Args>
 void **construct** (_Up * __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n) throw ()
- template<typename _Up >
 void **destroy** (_Up * __p)
- size_type **max_size** () const noexcept

Private Types

- typedef vector_type::iterator **iterator**
- typedef
 [__detail::__mini_vector](#)
 < value_type > **vector_type**

Private Member Functions

- void [_M_clear](#) ()
- size_t * [_M_get](#) (size_t __sz) throw (std::bad_alloc)
- void [_M_insert](#) (size_t * __addr) throw ()

4.31.1 Detailed Description

template<typename _Tp>class `__gnu_cxx::bitmap_allocator<_Tp>`

Bitmap Allocator, primary template.

Definition at line 687 of file `bitmap_allocator.h`.

4.31.2 Member Function Documentation

4.31.2.1 `template<typename _Tp> pointer __gnu_cxx::bitmap_allocator<_Tp>::M_allocate_single_object () throw (std::bad_alloc) [inline]`

Allocates memory for a single object of size `sizeof(_Tp)`.

Exceptions

<code>std::bad_alloc.</code>	If memory can not be allocated.
------------------------------	---------------------------------

Complexity: Worst case complexity is $O(N)$, but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of $O(1)$! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 827 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::_Bit_scan_forward()`.

4.31.2.2 `template<typename _Tp> void __gnu_cxx::bitmap_allocator<_Tp>::M_deallocate_single_object (pointer __p) throw () [inline]`

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity: $O(\lg(N))$, but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in $O(1)$ time by the `deallocate` function.

Definition at line 917 of file `bitmap_allocator.h`.

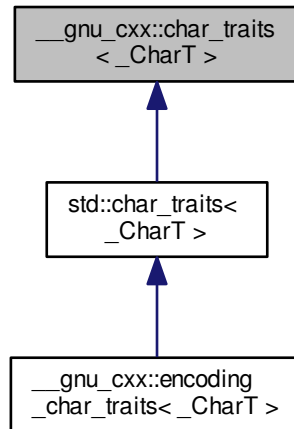
References `__gnu_cxx::__detail::__bit_free()`, `__gnu_cxx::__detail::__num_bitmaps()`, `std::__rotate()`, and `__gnu_cxx::free_list::_M_insert()`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.32 __gnu_cxx::char_traits<_CharT> Struct Template Reference

Inheritance diagram for __gnu_cxx::char_traits<_CharT>:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types<_CharT>::int_type` **int_type**
- typedef `_Char_types<_CharT>::off_type` **off_type**
- typedef `_Char_types<_CharT>::pos_type` **pos_type**
- typedef `_Char_types<_CharT>::state_type` **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type * __s, std::size_t __n, char_type __a)
- static int **compare** (const char_type * __s1, const char_type * __s2, std::size_t __n)
- static char_type * **copy** (char_type * __s1, const char_type * __s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type * __s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type * __s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type * __s1, const char_type * __s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)

- static constexpr `char_type` **to_char_type** (const `int_type` &__c)
- static constexpr `int_type` **to_int_type** (const `char_type` &__c)

4.32.1 Detailed Description

```
template<typename _CharT>struct __gnu_cxx::char_traits< _CharT >
```

Base class used to implement `std::char_traits`.

Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 83 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.33 `__gnu_cxx::character< V, I, S >` Struct Template Reference

Public Types

- typedef `character< V, I, S >` **char_type**
- typedef `I` **int_type**
- typedef `S` **state_type**
- typedef `V` **value_type**

Static Public Member Functions

- template<typename V2 >
static `char_type` **from** (const V2 &v)
- template<typename V2 >
static V2 **to** (const `char_type` &c)

Public Attributes

- `value_type` **value**

4.33.1 Detailed Description

```
template<typename V, typename I, typename S = std::mbstate_t>struct __gnu_cxx::character< V, I, S >
```

A POD class that serves as a character abstraction class.

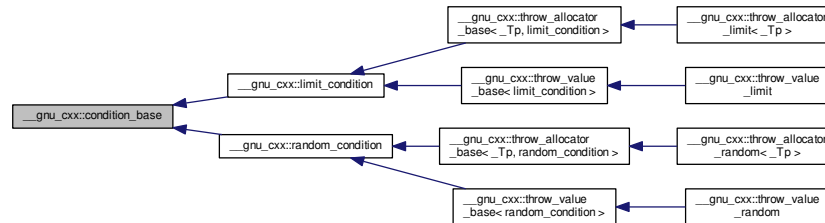
Definition at line 49 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

4.34 __gnu_cxx::condition_base Struct Reference

Inheritance diagram for __gnu_cxx::condition_base:



4.34.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature `void throw_conditionally()`

Definition at line 247 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.35 __gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2> Struct Template Reference

Inherits `__gnu_cxx::_Constant_binary_fun<_Result, _Arg1, _Arg2>`.

Public Types

- `typedef _Arg1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Arg2 second_argument_type`

Public Member Functions

- `constant_binary_fun` (const _Result &__v)
- `const result_type &operator()` (const _Arg1 &, const _Arg2 &) const

Public Attributes

- `_Result _M_val`

4.35.1 Detailed Description

```
template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 320 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.36 `__gnu_cxx::constant_unary_fun< _Result, _Argument >` Struct Template Reference

Inherits `__gnu_cxx::Constant_unary_fun< _Result, _Argument >`.

Public Types

- typedef `_Argument` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- **constant_unary_fun** (const `_Result` &__v)
- const `result_type` & **operator()** (const `_Argument` &) const

Public Attributes

- `result_type` **_M_val**

4.36.1 Detailed Description

```
template<class _Result, class _Argument = _Result>struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
```

An [SGI extension](#) .

Definition at line 312 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.37 `__gnu_cxx::constant_void_fun< _Result >` Struct Template Reference

Inherits `__gnu_cxx::Constant_void_fun< _Result >`.

Public Types

- typedef `_Result` **result_type**

Public Member Functions

- **constant_void_fun** (const _Result &__v)
- const result_type & **operator()** () const

Public Attributes

- result_type **_M_val**

4.37.1 Detailed Description

template<class _Result>struct `__gnu_cxx::constant_void_fun<_Result>`

An [SGI extension](#) .

Definition at line 303 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.38 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

Public Types

- typedef `_Alloc::const_pointer` **const_pointer**
- typedef `_Alloc::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `_Alloc::value_type` **value_type**

Public Member Functions

- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const void * __hint)
- void **deallocate** (pointer __p, size_type __n)

4.38.1 Detailed Description

template<typename _Alloc>class `__gnu_cxx::debug_allocator<_Alloc>`

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

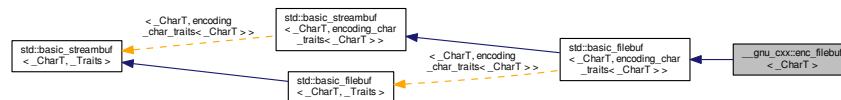
Definition at line 62 of file debug_allocator.h.

The documentation for this class was generated from the following file:

- [debug_allocator.h](#)

4.39 __gnu_cxx::enc_filebuf<_CharT> Class Template Reference

Inheritance diagram for __gnu_cxx::enc_filebuf<_CharT>:



Public Types

- typedef `codecvt< char_type, char, __state_type >` **__codecvt_type**
- typedef `__basic_file< char >` **__file_type**
- typedef `basic_filebuf< char_type, traits_type >` **__filebuf_type**
- typedef `traits_type::state_type` **__state_type**
- typedef `basic_streambuf< char_type, traits_type >` **__streambuf_type**
- typedef `_CharT` **char_type**
- typedef `traits_type::int_type` **int_type**
- typedef `traits_type::off_type` **off_type**
- typedef `traits_type::pos_type` **pos_type**
- typedef `traits_type::state_type` **state_type**
- typedef `encoding_char_traits<_CharT>` **traits_type**

Public Member Functions

- **enc_filebuf** (`state_type &__state`)
- `__filebuf_type * close` ()
- locale `getloc` () const
- streamsize `in_avail` ()
- bool `is_open` () const throw ()
- `__filebuf_type * open` (const char * __s, ios_base::openmode __mode)
- `__filebuf_type * open` (const std::string & __s, ios_base::openmode __mode)
- locale `pubimbue` (const locale & __loc)
- int_type `sputc` ()
- int_type `sgetc` ()
- streamsize `sgetn` (char_type * __s, streamsize __n)
- int_type `snextc` ()
- int_type `sputbackc` (char_type __c)

- `int_type sputc` (`char_type __c`)
- `streamsize sputn` (`const char_type *__s`, `streamsize __n`)
- `int_type sungetc` ()
- `basic_streambuf * pubsetbuf` (`char_type *__s`, `streamsize __n`)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()

Protected Member Functions

- `void __safe_gbump` (`streamsize __n`)
- `void __safe_pbump` (`streamsize __n`)
- `void _M_allocate_internal_buffer` ()
- `bool _M_convert_to_external` (`char_type *`, `streamsize`)
- `void _M_create_pback` ()
- `void _M_destroy_internal_buffer` () `throw ()`
- `void _M_destroy_pback` () `throw ()`
- `int _M_get_ext_pos` (`__state_type &__state`)
- `pos_type _M_seek` (`off_type __off`, `ios_base::seekdir __way`, `__state_type __state`)
- `void _M_set_buffer` (`streamsize __off`)
- `bool _M_terminate_output` ()
- `void gbump` (`int __n`)
- `virtual void imbue` (`const locale &__loc`)
- `virtual int_type overflow` (`int_type __c=encoding_char_traits<_CharT>::eof()`)
- `virtual int_type pbackfail` (`int_type __c=encoding_char_traits<_CharT>::eof()`)
- `void pbump` (`int __n`)
- `virtual pos_type seekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `virtual pos_type seekpos` (`pos_type __pos`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `virtual __streambuf_type * setbuf` (`char_type *__s`, `streamsize __n`)
- `void setg` (`char_type *__gbeg`, `char_type *__gnext`, `char_type *__gend`)
- `void setp` (`char_type *__pbeg`, `char_type *__pend`)
- `virtual streamsize showmanyc` ()
- `virtual int sync` ()
- `virtual int_type uflow` ()
- `virtual int_type underflow` ()
- `virtual streamsize xsgetn` (`char_type *__s`, `streamsize __n`)
- `virtual streamsize xsputn` (`const char_type *__s`, `streamsize __n`)
- `char_type * eback` () `const`
- `char_type * gptr` () `const`
- `char_type * egptr` () `const`
- `char_type * pbase` () `const`
- `char_type * pptr` () `const`
- `char_type * epptr` () `const`

Protected Attributes

- `char_type * _M_buf`
 - `bool _M_buf_allocated`
 - `locale _M_buf_locale`
 - `size_t _M_buf_size`
 - `const __codecvt_type * _M_codecvt`
 - `char * _M_ext_buf`
 - `streamsize _M_ext_buf_size`
 - `char * _M_ext_end`
 - `const char * _M_ext_next`
 - `__file_type _M_file`
 - `char_type * _M_in_beg`
 - `char_type * _M_in_cur`
 - `char_type * _M_in_end`
 - `__c_lock _M_lock`
 - `ios_base::openmode _M_mode`
 - `char_type * _M_out_beg`
 - `char_type * _M_out_cur`
 - `char_type * _M_out_end`
 - `bool _M_reading`
 - `__state_type _M_state_beg`
 - `__state_type _M_state_cur`
 - `__state_type _M_state_last`
 - `bool _M_writing`
-
- `char_type _M_pback`
 - `char_type * _M_pback_cur_save`
 - `char_type * _M_pback_end_save`
 - `bool _M_pback_init`

4.39.1 Detailed Description

```
template<typename _CharT> class __gnu_cxx::enc_filebuf<_CharT>
```

class `enc_filebuf`.

Definition at line 42 of file `enc_filebuf.h`.

4.39.2 Member Function Documentation

4.39.2.1 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M_create_pback () [inline], [protected], [inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 177 of file `fstream`.

4.39.2.2 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M_destroy_pback () throw () [inline], [protected], [inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 194 of file `fstream`.

4.39.2.3 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_set_buffer (streamsize __off)`
`[inline], [protected], [inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 395 of file `fstream`.

4.39.2.4 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::close ()` `[inherited]`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

4.39.2.5 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eback () const` `[inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 482 of file `streambuf`.

4.39.2.6 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr () const` `[inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 488 of file `streambuf`.

4.39.2.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eptr () const` `[inline]`, `[protected]`, `[inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 535 of file `streambuf`.

4.39.2.8 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gbump (int __n)` `[inline]`, `[protected]`, `[inherited]`

Moving the read position.

Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 498 of file `streambuf`.

4.39.2.9 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::getloc () const` `[inline]`, `[inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file `streambuf`.

4.39.2.10 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr () const` `[inline]`, `[protected]`, `[inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 485 of file `streambuf`.

4.39.2.11 `virtual void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue (const locale & __loc)`
`[protected], [virtual], [inherited]`

Changes translations.

Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

4.39.2.12 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::in_avail ()` `[inline], [inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file `streambuf`.

4.39.2.13 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::is_open () const throw ()` `[inline], [inherited]`

Returns true if the external file is open.

Definition at line 227 of file `fstream`.

4.39.2.14 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open (const char * __s, ios_base::openmode __mode)` `[inherited]`

Opens an external file.

Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +

ios_base Flag combination	stdio equivalent	binary	in	out	trunc	app
+ + w + + a + a + + w + r + + r+ + + + w+ + + + a+ + + a+ +						
+ + + wb + + + ab + + ab + + + wb + + rb + + + r+b + + + + w+b + + + + a+b + + + a+b +						

4.39.2.15 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open (const std::string & __s, ios_base::openmode __mode)` [inline], [inherited]

Opens an external file.

Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Returns

`this` on success, NULL on failure

Definition at line 280 of file `fstream`.

4.39.2.16 `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::overflow (int_type __c = Traits::eof())` [protected], [virtual], [inherited]

Consumes data from the buffer; writes to the controlled sequence.

Parameters

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

4.39.2.17 `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::pbackfail (int_type __c = Traits::eof())` [protected], [virtual], [inherited]

Tries to back up the input sequence.

Parameters

<code>__c</code>	The character to be inserted back into the sequence.
------------------	--

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

4.39.2.18 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbase () const`
`[inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 529 of file `streambuf`.

4.39.2.19 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbump (int __n)` `[inline],`
`[protected], [inherited]`

Moving the write position.

Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the write position without returning any data.

Definition at line 545 of file `streambuf`.

4.39.2.20 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pptr () const` `[inline],`
`[protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence

- `eptr()` returns the end pointer for the output sequence

Definition at line 532 of file `streambuf`.

4.39.2.21 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubimbue (const locale & __loc)`
`[inline], [inherited]`

Entry point for `imbue()`.

Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

Returns

The previous locale.

Calls the derived `imbue(__loc)`.

Definition at line 209 of file `streambuf`.

4.39.2.22 `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out)`
`[inline], [inherited]`

Alters the stream position.

Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekoff` function.

Definition at line 251 of file `streambuf`.

4.39.2.23 `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekpos (pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out)`
`[inline], [inherited]`

Alters the stream position.

Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 263 of file `streambuf`.

4.39.2.24 `basic_streambuf* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsetbuf (char_type* __s, streamsize __n)`
`[inline], [inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if

any) and returning the result unchanged.

Definition at line 239 of file `streambuf`.

4.39.2.25 `int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync () [inline], [inherited]`

Calls virtual sync function.

Definition at line 271 of file `streambuf`.

4.39.2.26 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sbumpc () [inline], [inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file `streambuf`.

4.39.2.27 `virtual pos_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::seekoff (off_type , ios_base::seekdir , ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

4.39.2.28 `virtual pos_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::seekpos (pos_type , ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

4.39.2.29 `virtual __streambuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::setbuf (char_type * __s, streamsize __n) [protected], [virtual], [inherited]`

Manipulates the buffer.

Parameters

<code>__s</code>	Pointer to a buffer area.
<code>__n</code>	Size of <code>__s</code> .

Returns

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.-html> for more.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

4.39.2.30 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)` [inline], [protected], [inherited]

Setting the three read area pointers.

Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 509 of file `streambuf`.

4.39.2.31 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setp (char_type * __pbeg, char_type * __pend)` [inline], [protected], [inherited]

Setting the three write area pointers.

Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 555 of file `streambuf`.

4.39.2.32 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetc ()` [inline], [inherited]

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file `streambuf`.

4.39.2.33 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetn (char_type * __s, streamsize __n)` `[inline]`, `[inherited]`

Entry point for `xsgetn`.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 357 of file `streambuf`.

4.39.2.34 `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::showmanyc ()` `[protected]`, `[virtual]`, `[inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

4.39.2.35 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::snextc ()` `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file `streambuf`.

4.39.2.36 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputback(char_type __c)`
`[inline], [inherited]`

Pushing characters back into the input stream.

Parameters

<code>__c</code>	The character to push back.
------------------	-----------------------------

Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 372 of file `streambuf`.

4.39.2.37 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::putc(char_type __c)`
`[inline], [inherited]`

Entry point for all single-character output functions.

Parameters

<code>__c</code>	A character to output.
------------------	------------------------

Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(-__c)`.

Definition at line 424 of file `streambuf`.

4.39.2.38 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputn(const char_type * __s, streamsize __n)`
`[inline], [inherited]`

Entry point for all single-character output functions.

Parameters

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 450 of file `streambuf`.

4.39.2.39 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sungetc ()` `[inline]`,
`[inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 397 of file `streambuf`.

4.39.2.40 `virtual int std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::sync (void)` `[protected]`,
`[virtual]`, `[inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

4.39.2.41 `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::uflow ()` `[inline]`,
`[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 700 of file `streambuf`.

4.39.2.42 `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::underflow ()`
`[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns eof().

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

4.39.2.43 virtual streamsize **std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsgetn** (char_type * __s, streamsize __n) [protected], [virtual], [inherited]

Multiple character extraction.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

4.39.2.44 virtual streamsize **std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsputn** (const char_type * __s, streamsize __n) [protected], [virtual], [inherited]

Multiple character insertion.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>](#).

4.39.3 Member Data Documentation

4.39.3.1 `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf` [protected],
[inherited]

Pointer to the beginning of internal buffer.

Definition at line 114 of file `fstream`.

4.39.3.2 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf.locale` [protected],
[inherited]

Current locale setting.

Definition at line 192 of file `streambuf`.

4.39.3.3 `size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf.size` [protected],
[inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 121 of file `fstream`.

4.39.3.4 `char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf` [protected],
[inherited]

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 156 of file `fstream`.

4.39.3.5 `streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf.size` [protected],
[inherited]

Size of buffer held by `_M_ext_buf`.

Definition at line 161 of file `fstream`.

4.39.3.6 `const char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_next` [protected],
[inherited]

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 168 of file `fstream`.

4.39.3.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_beg` [protected],
[inherited]

Start of get area.

Definition at line 184 of file `streambuf`.

4.39.3.8 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_cur` [protected],
[inherited]

Current read area.

Definition at line 185 of file `streambuf`.

4.39.3.9 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::M.in_end` [protected], [inherited]

End of get area.

Definition at line 186 of file `streambuf`.

4.39.3.10 `ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M.mode` [protected], [inherited]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 99 of file `fstream`.

4.39.3.11 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::M.out_beg` [protected], [inherited]

Start of put area.

Definition at line 187 of file `streambuf`.

4.39.3.12 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::M.out_cur` [protected], [inherited]

Current put area.

Definition at line 188 of file `streambuf`.

4.39.3.13 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::M.out_end` [protected], [inherited]

End of put area.

Definition at line 189 of file `streambuf`.

4.39.3.14 `char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M.pback` [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 142 of file `fstream`.

4.39.3.15 `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M.pback_cur_save` [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 143 of file `fstream`.

4.39.3.16 `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M.pback_end_save` [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 144 of file `fstream`.

4.39.3.17 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M_pback_init` `[protected]`,
`[inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 145 of file `fstream`.

4.39.3.18 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::M_reading` `[protected]`,
`[inherited]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

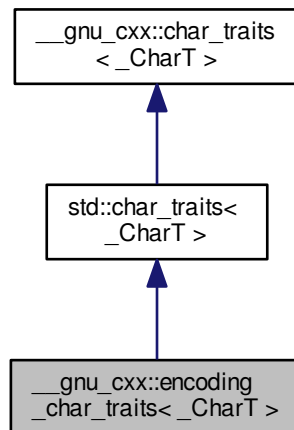
Definition at line 133 of file `fstream`.

The documentation for this class was generated from the following file:

- [enc_filebuf.h](#)

4.40 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::encoding_char_traits<_CharT>`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types<_CharT>` `::int_type` **int_type**
- typedef `_Char_types<_CharT>` `::off_type` **off_type**
- typedef `std::fpos<state_type>` **pos_type**
- typedef `encoding_state` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **eof** ()
- static constexpr bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static constexpr bool **eq_int_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)
- static `std::size_t` **length** (`const char_type *__s`)
- static constexpr bool **lt** (`const char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **not_eof** (`const int_type &__c`)
- static constexpr `char_type` **to_char_type** (`const int_type &__c`)
- static constexpr `int_type` **to_int_type** (`const char_type &__c`)

4.40.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::encoding_char_traits<_CharT>`

`encoding_char_traits`

Definition at line 210 of file `codecvt_specializations.h`.

The documentation for this struct was generated from the following file:

- [codecvt_specializations.h](#)

4.41 `__gnu_cxx::encoding_state` Class Reference

Public Types

- typedef `iconv_t` **descriptor_type**

Public Member Functions

- **encoding_state** (`const char *__int`, `const char *__ext`, `int __ibom=0`, `int __ebom=0`, `int __bytes=1`)
- **encoding_state** (`const encoding_state &__obj`)
- int **character_ratio** () const
- int **external_bom** () const

- const `std::string` `external_encoding` () const
- bool `good` () const throw ()
- const descriptor_type & `in_descriptor` () const
- int `internal_bom` () const
- const `std::string` `internal_encoding` () const
- `encoding_state` & `operator=` (const `encoding_state` &__obj)
- const descriptor_type & `out_descriptor` () const

Protected Member Functions

- void `construct` (const `encoding_state` &__obj)
- void `destroy` () throw ()
- void `init` ()

Protected Attributes

- int `_M_bytes`
- int `_M_ext_bom`
- `std::string` `_M_ext_enc`
- descriptor_type `_M_in_desc`
- int `_M_int_bom`
- `std::string` `_M_int_enc`
- descriptor_type `_M_out_desc`

4.41.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

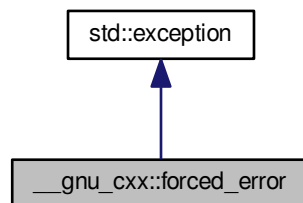
Definition at line 50 of file `codecvt_specializations.h`.

The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

4.42 `__gnu_cxx::forced_error` Struct Reference

Inheritance diagram for `__gnu_cxx::forced_error`:



Public Member Functions

- virtual const char * [what](#) () const noexcept

4.42.1 Detailed Description

Thrown by exception safety machinery.

Definition at line 74 of file `throw_allocator.h`.

4.42.2 Member Function Documentation

4.42.2.1 virtual const char* `std::exception::what` () const [virtual],[noexcept],[inherited]

Returns a C-style character string describing the general cause of the current error.

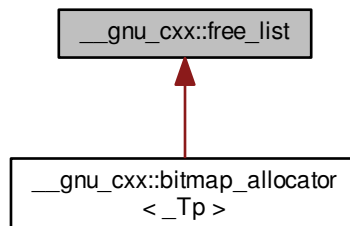
Reimplemented in `std::bad_function_call`, `std::ios_base::failure`, `std::bad_typeid`, `std::bad_cast`, `std::runtime_error`, `std::future_error`, `std::bad_exception`, `std::logic_error`, `std::bad_weak_ptr`, and `std::bad_alloc`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.43 `__gnu_cxx::free_list` Class Reference

Inheritance diagram for `__gnu_cxx::free_list`:



Public Types

- typedef `__mutex` **`__mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t *` **`value_type`**
- typedef
 [__detail::__mini_vector](#)
 `< value_type >` **`vector_type`**

Public Member Functions

- void `_M_clear` ()
- `size_t * _M_get` (`size_t __sz`) throw (`std::bad_alloc`)
- void `_M_insert` (`size_t * __addr`) throw ()

4.43.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

Definition at line 521 of file `bitmap_allocator.h`.

4.43.2 Member Function Documentation

4.43.2.1 void `__gnu_cxx::free_list::_M_clear` ()

This function just clears the internal Free List, and gives back all the memory to the OS.

4.43.2.2 `size_t * __gnu_cxx::free_list::_M_get` (`size_t __sz`) throw (`std::bad_alloc`)

This function gets a block of memory of the specified size from the free list.

Parameters

<code>__sz</code>	The size in bytes of the memory required.
-------------------	---

Returns

A pointer to the new memory block of size at least equal to that requested.

4.43.2.3 void `__gnu_cxx::free_list::_M_insert` (`size_t * __addr`) throw () [inline]

This function returns the block of memory to the internal free list.

Parameters

<code>__addr</code>	The pointer to the memory block that was given by a call to the <code>_M_get</code> function.
---------------------	---

Definition at line 631 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.44 `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference

Public Types

- typedef `_Ht::allocator_type` **allocator_type**
- typedef `_Ht::const_iterator` **const_iterator**
- typedef `_Ht::const_pointer` **const_pointer**

- typedef `_Ht::const_reference` `const_reference`
- typedef `_Tp` `data_type`
- typedef `_Ht::difference_type` `difference_type`
- typedef `_Ht::hasher` `hasher`
- typedef `_Ht::iterator` `iterator`
- typedef `_Ht::key_equal` `key_equal`
- typedef `_Ht::key_type` `key_type`
- typedef `_Tp` `mapped_type`
- typedef `_Ht::pointer` `pointer`
- typedef `_Ht::reference` `reference`
- typedef `_Ht::size_type` `size_type`
- typedef `_Ht::value_type` `value_type`

Public Member Functions

- `hash_map` (`size_type __n`)
- `hash_map` (`size_type __n`, `const hasher &__hf`)
- `hash_map` (`size_type __n`, `const hasher &__hf`, `const key_equal &__eq`, `const allocator_type &__a=allocator_type()`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f`, `_InputIterator __l`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n`, `const hasher &__hf`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n`, `const hasher &__hf`, `const key_equal &__eq`, `const allocator_type &__a=allocator_type()`)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `size_type bucket_count` () `const`
- `void clear` ()
- `size_type count` (`const key_type &__key`) `const`
- `size_type elems_in_bucket` (`size_type __n`) `const`
- `bool empty` () `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `pair< iterator, iterator> equal_range` (`const key_type &__key`)
- `pair< const_iterator, const_iterator> equal_range` (`const key_type &__key`) `const`
- `size_type erase` (`const key_type &__key`)
- `void erase` (`iterator __it`)
- `void erase` (`iterator __f`, `iterator __l`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `allocator_type get_allocator` () `const`
- `hasher hash_func` () `const`
- `pair< iterator, bool> insert` (`const value_type &__obj`)
- `template<class _InputIterator>`
`void insert` (`_InputIterator __f`, `_InputIterator __l`)
- `pair< iterator, bool> insert_noresize` (`const value_type &__obj`)

- `key_equal` **key_eq** () const
- `size_type` **max_bucket_count** () const
- `size_type` **max_size** () const
- `_Tp` & **operator[]** (const `key_type` &__key)
- void **resize** (size_type __hint)
- `size_type` **size** () const
- void **swap** ([hash_map](#) &__hs)

Friends

- `template<class _K1, class _T1, class _HF, class _EqK, class _Al>`
`bool operator==` (const [hash_map](#)<_K1, _T1, _HF, _EqK, _Al> &, const [hash_map](#)<_K1, _T1, _HF, _EqK, _Al> &)

4.44.1 Detailed Description

`template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>class __gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Definition at line 83 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash_map](#)

4.45 `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference

Public Types

- `typedef _Ht::allocator_type` **allocator_type**
- `typedef _Ht::const_iterator` **const_iterator**
- `typedef _Ht::const_pointer` **const_pointer**
- `typedef _Ht::const_reference` **const_reference**
- `typedef _Tp` **data_type**
- `typedef _Ht::difference_type` **difference_type**
- `typedef _Ht::hasher` **hasher**
- `typedef _Ht::iterator` **iterator**
- `typedef _Ht::key_equal` **key_equal**
- `typedef _Ht::key_type` **key_type**
- `typedef _Tp` **mapped_type**
- `typedef _Ht::pointer` **pointer**
- `typedef _Ht::reference` **reference**
- `typedef _Ht::size_type` **size_type**
- `typedef _Ht::value_type` **value_type**

Public Member Functions

- **hash_multimap** (size_type __n)
- **hash_multimap** (size_type __n, const hasher &__hf)
- **hash_multimap** (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- template<class __InputIterator >
hash_multimap (__InputIterator __f, __InputIterator __l)
- template<class __InputIterator >
hash_multimap (__InputIterator __f, __InputIterator __l, size_type __n)
- template<class __InputIterator >
hash_multimap (__InputIterator __f, __InputIterator __l, size_type __n, const hasher &__hf)
- template<class __InputIterator >
hash_multimap (__InputIterator __f, __InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __it)
- void **erase** (iterator __f, iterator __l)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- iterator **insert** (const value_type &__obj)
- template<class __InputIterator >
void **insert** (__InputIterator __f, __InputIterator __l)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multimap &__hs)

Friends

- template<class __K1, class __T1, class __HF, class __EqK, class __AI >
bool **operator==** (const hash_multimap< __K1, __T1, __HF, __EqK, __AI > &, const hash_multimap< __K1, __T1, __HF, __EqK, __AI > &)

4.45.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>>class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Definition at line 296 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash_map](#)

4.46 `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- `hash_multiset (size_type __n)`
- `hash_multiset (size_type __n, const hasher &__hf)`
- `hash_multiset (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `iterator begin () const`
- `size_type bucket_count () const`

- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __it)
- void **erase** (iterator __f, iterator __l)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- iterator **insert** (const value_type &__obj)
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_multiset](#) &hs)

Friends

- template<class _Val, class _HF, class _EqK, class _AI >
bool **operator==** (const [hash_multiset](#)< _Val, _HF, _EqK, _AI > &, const [hash_multiset](#)< _Val, _HF, _EqK, _AI > &)

4.46.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Definition at line 285 of file `hash_set`.

The documentation for this class was generated from the following file:

- [hash_set](#)

4.47 `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- `hash_set (size_type __n)`
- `hash_set (size_type __n, const hasher &__hf)`
- `hash_set (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`
- `template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `iterator begin () const`
- `size_type bucket_count () const`
- `void clear ()`
- `size_type count (const key_type &__key) const`
- `size_type elems_in_bucket (size_type __n) const`
- `bool empty () const`
- `iterator end () const`
- `pair< iterator, iterator > equal_range (const key_type &__key) const`
- `size_type erase (const key_type &__key)`
- `void erase (iterator __it)`
- `void erase (iterator __f, iterator __l)`
- `iterator find (const key_type &__key) const`
- `allocator_type get_allocator () const`
- `hasher hash_funct () const`
- `pair< iterator, bool > insert (const value_type &__obj)`
- `template<class _InputIterator >
void insert (_InputIterator __f, _InputIterator __l)`
- `pair< iterator, bool > insert_noresize (const value_type &__obj)`

- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_set &__hs)

Friends

- template<class _Val, class _HF, class _EqK, class _AI >
bool **operator==** (const hash_set< _Val, _HF, _EqK, _AI > &, const hash_set< _Val, _HF, _EqK, _AI > &)

4.47.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

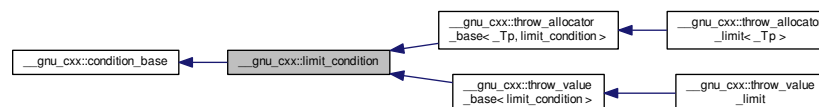
Definition at line 84 of file hash_set.

The documentation for this class was generated from the following file:

- [hash_set](#)

4.48 __gnu_cxx::limit_condition Struct Reference

Inheritance diagram for __gnu_cxx::limit_condition:



Classes

- struct [always_adjustor](#)
Always enter the condition.
- struct [limit_adjustor](#)
Enter the nth condition.
- struct [never_adjustor](#)
Never enter the condition.

Static Public Member Functions

- static `size_t & count` ()
- static `size_t & limit` ()
- static void `set_limit` (const `size_t` __l)
- static void `throw_conditionally` ()

4.48.1 Detailed Description

Base class for incremental control and throw.

Definition at line 256 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.49 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

4.49.1 Detailed Description

Always enter the condition.

Definition at line 280 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.50 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

Public Member Functions

- `limit_adjustor` (const `size_t` __l)

4.50.1 Detailed Description

Enter the nth condition.

Definition at line 286 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.51 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

4.51.1 Detailed Description

Never enter the condition.

Definition at line 274 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.52 `__gnu_cxx::malloc_allocator< _Tp >` Class Template Reference

Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef [std::true_type](#) **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **malloc_allocator** (const [malloc_allocator](#) &) noexcept
- template<typename _Tp1 >
 malloc_allocator (const [malloc_allocator](#)< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *==0)
- template<typename _Up , typename... _Args>
 void **construct** (_Up * __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type)
- template<typename _Up >
 void **destroy** (_Up * __p)
- size_type **max_size** () const noexcept

4.52.1 Detailed Description

template<typename _Tp>class `__gnu_cxx::malloc_allocator< _Tp >`

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

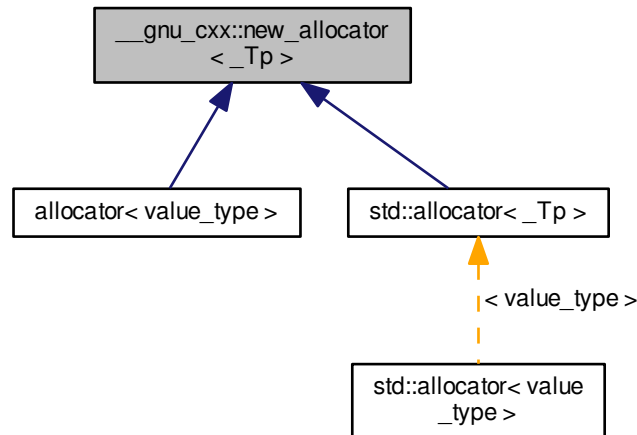
Definition at line 56 of file `malloc_allocator.h`.

The documentation for this class was generated from the following file:

- [malloc_allocator.h](#)

4.53 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



Public Types

- typedef const `_Tp` * **const_pointer**
- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef `_Tp` * **pointer**
- typedef [std::true_type](#) **propagate_on_container_move_assignment**
- typedef `_Tp` & **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **new_allocator** (const [new_allocator](#) &) noexcept
- template<typename `_Tp1` >
 new_allocator (const [new_allocator](#)<`_Tp1` > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *==0)
- template<typename `_Up`, typename... `_Args`>
 void **construct** (`_Up` * __p, `_Args` &&... __args)
- void **deallocate** (pointer __p, size_type)
- template<typename `_Up` >
 void **destroy** (`_Up` * __p)
- size_type **max_size** () const noexcept

4.53.1 Detailed Description

```
template<typename _Tp>class __gnu_cxx::new_allocator< _Tp >
```

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 58 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new_allocator.h](#)

4.54 `__gnu_cxx::project1st< _Arg1, _Arg2 >` Struct Template Reference

Inherits `__gnu_cxx::Project1st< _Arg1, _Arg2 >`.

Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

Public Member Functions

- `_Arg1` **operator()** (const `_Arg1` &__x, const `_Arg2` &) const

4.54.1 Detailed Description

```
template<class _Arg1, class _Arg2>struct __gnu_cxx::project1st< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 237 of file `ext/functional`.

4.54.2 Member Typedef Documentation

4.54.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.54.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.54.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.55 `__gnu_cxx::project2nd<_Arg1, _Arg2>` Struct Template Reference

Inherits `__gnu_cxx::Project2nd<_Arg1, _Arg2>`.

Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

Public Member Functions

- `_Arg2 operator()` (const `_Arg1` &, const `_Arg2` &__y) const

4.55.1 Detailed Description

`template<class _Arg1, class _Arg2> struct __gnu_cxx::project2nd<_Arg1, _Arg2>`

An [SGI extension](#) .

Definition at line 241 of file `ext/functional`.

4.55.2 Member Typedef Documentation

4.55.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.55.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.55.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

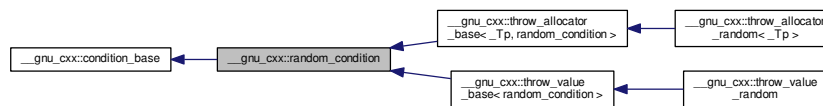
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.56 __gnu_cxx::random_condition Struct Reference

Inheritance diagram for `__gnu_cxx::random_condition`:



Classes

- struct [always_adjustor](#)
Always enter the condition.
- struct [group_adjustor](#)
Group condition.
- struct [never_adjustor](#)
Never enter the condition.

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

4.56.1 Detailed Description

Base class for random probability control and throw.

Definition at line 328 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.57 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

4.57.1 Detailed Description

Always enter the condition.

Definition at line 361 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.58 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

Public Member Functions

- `group_adjustor` (`size_t` size)

4.58.1 Detailed Description

Group condition.

Definition at line 346 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.59 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

4.59.1 Detailed Description

Never enter the condition.

Definition at line 355 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.60 `__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc >`.

Public Types

- `typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef const _Rb_tree_node< _Val > * _Const_Link_type`
- `typedef _Rb_tree_node< _Val > * _Link_type`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rb_tree_iterator< value_type > iterator`
- `typedef _Key key_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _Val value_type`

Public Member Functions

- `rb_tree (const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`
- `bool __rb_verify () const`
- `template<typename... _Args> iterator _M_emplace_equal (_Args &&...__args)`
- `template<typename... _Args> iterator _M_emplace_hint_equal (const_iterator __pos, _Args &&...__args)`
- `template<typename... _Args> iterator _M_emplace_hint_unique (const_iterator __pos, _Args &&...__args)`
- `template<typename... _Args> pair< iterator, bool > _M_emplace_unique (_Args &&...__args)`
- `_Node_allocator & _M_get_Node_allocator () noexcept`
- `const _Node_allocator & _M_get_Node_allocator () const noexcept`
- `template<typename _Arg > iterator _M_insert_equal (_Arg &&__x)`
- `template<typename _InputIterator > void _M_insert_equal (_InputIterator __first, _InputIterator __last)`
- `template<class _II > void _M_insert_equal (_II __first, _II __last)`
- `template<typename _Arg > iterator _M_insert_equal_ (const_iterator __position, _Arg &&__x)`
- `template<typename _Arg > pair< iterator, bool > _M_insert_unique (_Arg &&__x)`
- `template<typename _InputIterator > void _M_insert_unique (_InputIterator __first, _InputIterator __last)`

- `template<class _II >`
`void M_insert_unique (_II __first, _II __last)`
- `template<typename _Arg >`
`iterator M_insert_unique (const_iterator __position, _Arg &&__x)`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `void clear () noexcept`
- `size_type count (const key_type &__k) const`
- `bool empty () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `pair< iterator, iterator > equal_range (const key_type &__k)`
- `pair< const_iterator,`
`const_iterator > equal_range (const key_type &__k) const`
- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`
- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `void erase (const key_type *__first, const key_type *__last)`
- `iterator find (const key_type &__k)`
- `const_iterator find (const key_type &__k) const`
- `allocator_type get_allocator () const noexcept`
- `_Compare key_comp () const`
- `iterator lower_bound (const key_type &__k)`
- `const_iterator lower_bound (const key_type &__k) const`
- `size_type max_size () const noexcept`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `size_type size () const noexcept`
- `void swap (_Rb_tree &__t)`
- `iterator upper_bound (const key_type &__k)`
- `const_iterator upper_bound (const key_type &__k) const`

Protected Types

- `typedef _Rb_tree_node_base * _Base_ptr`
- `typedef const _Rb_tree_node_base * _Const_Base_ptr`

Protected Member Functions

- `_Link_type M_begin ()`
- `_Const_Link_type M_begin () const`
- `_Link_type M_clone_node (_Const_Link_type __x)`
- `template<typename... _Args>`
`_Link_type M_create_node (_Args &&...__args)`
- `void M_destroy_node (_Link_type __p)`
- `_Link_type M_end ()`
- `_Const_Link_type M_end () const`

- `_Link_type _M_get_node ()`
- `_Base_ptr & _M_leftmost ()`
- `_Const_Base_ptr _M_leftmost () const`
- `void _M_put_node (_Link_type __p)`
- `_Base_ptr & _M_rightmost ()`
- `_Const_Base_ptr _M_rightmost () const`
- `_Base_ptr & _M_root ()`
- `_Const_Base_ptr _M_root () const`

Static Protected Member Functions

- `static const _Key & _S_key (_Const_Link_type __x)`
- `static const _Key & _S_key (_Const_Base_ptr __x)`
- `static _Link_type _S_left (_Base_ptr __x)`
- `static _Const_Link_type _S_left (_Const_Base_ptr __x)`
- `static _Base_ptr _S_maximum (_Base_ptr __x)`
- `static _Const_Base_ptr _S_maximum (_Const_Base_ptr __x)`
- `static _Base_ptr _S_minimum (_Base_ptr __x)`
- `static _Const_Base_ptr _S_minimum (_Const_Base_ptr __x)`
- `static _Link_type _S_right (_Base_ptr __x)`
- `static _Const_Link_type _S_right (_Const_Base_ptr __x)`
- `static const_reference _S_value (_Const_Link_type __x)`
- `static const_reference _S_value (_Const_Base_ptr __x)`

Protected Attributes

- `_Rb_tree_impl<_Compare> _M_impl`

4.60.1 Detailed Description

`template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = allocator<_Value>> struct __gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc>`

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

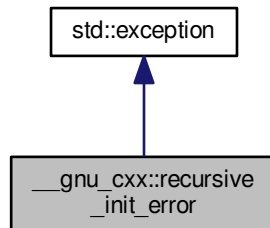
Definition at line 80 of file `rb_tree`.

The documentation for this struct was generated from the following file:

- `rb_tree`

4.61 `__gnu_cxx::recursive_init_error` Class Reference

Inheritance diagram for `__gnu_cxx::recursive_init_error`:



Public Member Functions

- virtual const char * [what](#) () const noexcept

4.61.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 689 of file `cxxabi.h`.

4.61.2 Member Function Documentation

4.61.2.1 virtual const char* `std::exception::what` () const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_function_call](#), [std::ios_base::failure](#), [std::bad_typeid](#), [std::bad_cast](#), [std::runtime_error](#), [std::future_error](#), [std::bad_exception](#), [std::logic_error](#), [std::bad_weak_ptr](#), and [std::bad_alloc](#).

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

4.62 `__gnu_cxx::rope<_CharT, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::_Rope_base<_CharT, _Alloc>`.

Public Types

- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` **__C**
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` **__F**
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` **__L**
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` **__S**
- typedef `_Alloc::template rebind<__C>::other` **_CAlloc**
- typedef `_Alloc::template rebind<_CharT>::other` **_DataAlloc**
- typedef `_Alloc::template rebind<__F>::other` **_FAlloc**
- typedef `_Alloc::template rebind<__L>::other` **_LAlloc**
- typedef `_Alloc::template rebind<__S>::other` **_SAlloc**
- typedef `_Rope_const_iterator<_CharT, _Alloc>` **const_iterator**
- typedef `const _CharT *` **const_pointer**
- typedef `_CharT` **const_reference**
- typedef `std::reverse_iterator<const_iterator>` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Rope_iterator<_CharT, _Alloc>` **iterator**
- typedef `_Rope_char_ptr_proxy<_CharT, _Alloc>` **pointer**
- typedef `_Rope_char_ref_proxy<_CharT, _Alloc>` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_CharT` **value_type**

Public Member Functions

- **rope** (`const _CharT * __s, const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s, size_t __len, const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s, const _CharT * __e, const allocator_type & __a=allocator_type()`)
- **rope** (`const const_iterator & __s, const const_iterator & __e, const allocator_type & __a=allocator_type()`)
- **rope** (`const iterator & __s, const iterator & __e, const allocator_type & __a=allocator_type()`)
- **rope** (`_CharT __c, const allocator_type & __a=allocator_type()`)
- **rope** (`size_t __n, _CharT __c, const allocator_type & __a=allocator_type()`)
- **rope** (`const allocator_type & __a=allocator_type()`)
- **rope** (`char_producer<_CharT> * __fn, size_t __len, bool __delete_fn, const allocator_type & __a=allocator_type()`)
- **rope** (`const rope & __x, const allocator_type & __a=allocator_type()`)

- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- `rope & append (const _CharT * __iter, size_t __n)`
- `rope & append (const _CharT * __c_string)`
- `rope & append (const _CharT * __s, const _CharT * __e)`
- `rope & append (const_iterator __s, const_iterator __e)`
- `rope & append (_CharT __c)`
- `rope & append ()`
- `rope & append (const rope & __y)`
- `rope & append (size_t __n, _CharT __c)`
- `void apply_to_pieces (size_t __begin, size_t __end, _Rope_char_consumer<_CharT> & __c) const`
- `_CharT at (size_type __pos) const`
- `_CharT back () const`
- `void balance ()`
- `const_iterator begin () const`
- `const_iterator begin ()`
- `const _CharT * c_str () const`
- `void clear ()`
- `int compare (const rope & __y) const`
- `const_iterator const_begin () const`
- `const_iterator const_end () const`
- `const_reverse_iterator const_rbegin () const`
- `const_reverse_iterator const_rend () const`
- `void copy (_CharT * __buffer) const`
- `size_type copy (size_type __pos, size_type __n, _CharT * __buffer) const`
- `void delete_c_str ()`
- `void dump ()`
- `bool empty () const`
- `const_iterator end () const`
- `const_iterator end ()`
- `void erase (size_t __p, size_t __n)`
- `void erase (size_t __p)`
- `iterator erase (const iterator & __p, const iterator & __q)`
- `iterator erase (const iterator & __p)`
- `size_type find (_CharT __c, size_type __pos=0) const`
- `size_type find (const _CharT * __s, size_type __pos=0) const`
- `_CharT front () const`
- `allocator_type get_allocator () const`
- `void insert (size_t __p, const rope & __r)`
- `void insert (size_t __p, size_t __n, _CharT __c)`
- `void insert (size_t __p, const _CharT * __i, size_t __n)`
- `void insert (size_t __p, const _CharT * __c_string)`
- `void insert (size_t __p, _CharT __c)`
- `void insert (size_t __p)`
- `void insert (size_t __p, const _CharT * __i, const _CharT * __j)`
- `void insert (size_t __p, const const_iterator & __i, const const_iterator & __j)`
- `void insert (size_t __p, const iterator & __i, const iterator & __j)`
- `iterator insert (const iterator & __p, const rope & __r)`
- `iterator insert (const iterator & __p, size_t __n, _CharT __c)`
- `iterator insert (const iterator & __p, _CharT __c)`
- `iterator insert (const iterator & __p)`

- iterator **insert** (const iterator &__p, const _CharT *c_string)
- iterator **insert** (const iterator &__p, const _CharT *__i, size_t __n)
- iterator **insert** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- iterator **insert** (const iterator &__p, const const_iterator &__i, const const_iterator &__j)
- iterator **insert** (const iterator &__p, const iterator &__i, const iterator &__j)
- size_type **length** () const
- size_type **max_size** () const
- iterator **mutable_begin** ()
- iterator **mutable_end** ()
- [reverse_iterator](#) **mutable_rbegin** ()
- reference **mutable_reference_at** (size_type __pos)
- [reverse_iterator](#) **mutable_rend** ()
- [rope](#) & **operator=** (const [rope](#) &__x)
- _CharT **operator[]** (size_type __pos) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_CharT __x)
- void **push_front** (_CharT __x)
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rend** () const
- [const_reverse_iterator](#) **rend** ()
- void **replace** (size_t __p, size_t __n, const [rope](#) &__r)
- void **replace** (size_t __p, size_t __n, const _CharT *__i, size_t __i_len)
- void **replace** (size_t __p, size_t __n, _CharT __c)
- void **replace** (size_t __p, size_t __n, const _CharT *__c_string)
- void **replace** (size_t __p, size_t __n, const _CharT *__i, const _CharT *__j)
- void **replace** (size_t __p, size_t __n, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, size_t __n, const iterator &__i, const iterator &__j)
- void **replace** (size_t __p, _CharT __c)
- void **replace** (size_t __p, const [rope](#) &__r)
- void **replace** (size_t __p, const _CharT *__i, size_t __i_len)
- void **replace** (size_t __p, const _CharT *__c_string)
- void **replace** (size_t __p, const _CharT *__i, const _CharT *__j)
- void **replace** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, const iterator &__i, const iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const [rope](#) &__r)
- void **replace** (const iterator &__p, const iterator &__q, _CharT __c)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__c_string)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, size_t __n)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const const_iterator &__i, const const_iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const iterator &__i, const iterator &__j)
- void **replace** (const iterator &__p, const [rope](#) &__r)
- void **replace** (const iterator &__p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__c_string)
- void **replace** (const iterator &__p, const _CharT *__i, size_t __n)
- void **replace** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const_iterator __i, const_iterator __j)
- void **replace** (const iterator &__p, iterator __i, iterator __j)
- const _CharT * **replace_with_c_str** ()

- `size_type size () const`
- `rope substr (size_t __start, size_t __len=1) const`
- `rope substr (iterator __start, iterator __end) const`
- `rope substr (iterator __start) const`
- `rope substr (const_iterator __start, const_iterator __end) const`
- `rope<_CharT, _Alloc> substr (const_iterator __start)`
- `void swap (rope &__b)`

Static Public Member Functions

- `static __C * _C_allocate (size_t __n)`
- `static void _C_deallocate (__C * __p, size_t __n)`
- `static _CharT * _Data_allocate (size_t __n)`
- `static void _Data_deallocate (_CharT * __p, size_t __n)`
- `static __F * _F_allocate (size_t __n)`
- `static void _F_deallocate (__F * __p, size_t __n)`
- `static __L * _L_allocate (size_t __n)`
- `static void _L_deallocate (__L * __p, size_t __n)`
- `static __S * _S_allocate (size_t __n)`
- `static void _S_deallocate (__S * __p, size_t __n)`

Public Attributes

- `_RopeRep * _M_tree_ptr`

Static Public Attributes

- `static const size_type npos`

Protected Types

- `enum { _S_copy_max }`
- `typedef _Rope_base<_CharT, _Alloc> _Base`
- `typedef _CharT * _Cstrptr`
- `typedef _Rope_RopeConcatenation<_CharT, _Alloc> _RopeConcatenation`
- `typedef _Rope_RopeFunction<_CharT, _Alloc> _RopeFunction`
- `typedef _Rope_RopeLeaf<_CharT, _Alloc> _RopeLeaf`
- `typedef _Rope_RopeRep<_CharT, _Alloc> _RopeRep`
- `typedef _Rope_RopeSubstring<_CharT, _Alloc> _RopeSubstring`
- `typedef _Rope_self_destruct_ptr<_CharT, _Alloc> _Self_destruct_ptr`
- `typedef _Base::allocator_type allocator_type`

Static Protected Member Functions

- static `size_t` **S_allocated_capacity** (`size_t` __n)
- static `bool` **S_apply_to_pieces** (`_Rope_char_consumer<_CharT>` &__c, `const _RopeRep *` __r, `size_t` __begin, `size_t` __end)
- static `_RopeRep *` **S_concat** (`_RopeRep *` __left, `_RopeRep *` __right)
- static `_RopeRep *` **S_concat_char_iter** (`_RopeRep *` __r, `const _CharT *` __iter, `size_t` __slen)
- static `_RopeRep *` **S_destr_concat_char_iter** (`_RopeRep *` __r, `const _CharT *` __iter, `size_t` __slen)
- static `_RopeLeaf *` **S_destr_leaf_concat_char_iter** (`_RopeLeaf *` __r, `const _CharT *` __iter, `size_t` __slen)
- static `_CharT` **S_fetch** (`_RopeRep *` __r, `size_type` __pos)
- static `_CharT *` **S_fetch_ptr** (`_RopeRep *` __r, `size_type` __pos)
- static `bool` **S_is0** (`_CharT` __c)
- static `_RopeLeaf *` **S_leaf_concat_char_iter** (`_RopeLeaf *` __r, `const _CharT *` __iter, `size_t` __slen)
- static `_RopeConcatenation *` **S_new_RopeConcatenation** (`_RopeRep *` __left, `_RopeRep *` __right, `allocator_type` &__a)
- static `_RopeFunction *` **S_new_RopeFunction** (`char_producer<_CharT>` *__f, `size_t` __size, `bool` __d, `allocator_type` &__a)
- static `_RopeLeaf *` **S_new_RopeLeaf** (`_CharT *` __s, `size_t` __size, `allocator_type` &__a)
- static `_RopeSubstring *` **S_new_RopeSubstring** (`_Rope_RopeRep<_CharT, _Alloc>` *__b, `size_t` __s, `size_t` __l, `allocator_type` &__a)
- static `void` **S_ref** (`_RopeRep *` __t)
- static `_RopeLeaf *` **S_RopeLeaf_from_unowned_char_ptr** (`const _CharT *` __s, `size_t` __size, `allocator_type` &__a)
- static `size_t` **S_rounded_up_size** (`size_t` __n)
- static `_RopeRep *` **S_substring** (`_RopeRep *` __base, `size_t` __start, `size_t` __endp1)
- static `_RopeRep *` **S_tree_concat** (`_RopeRep *` __left, `_RopeRep *` __right)
- static `void` **S_unref** (`_RopeRep *` __t)
- static `_RopeRep *` **replace** (`_RopeRep *` __old, `size_t` __pos1, `size_t` __pos2, `_RopeRep *` __r)

Static Protected Attributes

- static `_CharT` **S_empty_c_str** [1]

Friends

- class `_Rope_char_ptr_proxy<_CharT, _Alloc>`
- class `_Rope_char_ref_proxy<_CharT, _Alloc>`
- class `_Rope_const_iterator<_CharT, _Alloc>`
- class `_Rope_iterator<_CharT, _Alloc>`
- class `_Rope_iterator_base<_CharT, _Alloc>`
- struct `_Rope_RopeRep<_CharT, _Alloc>`
- struct `_Rope_RopeSubstring<_CharT, _Alloc>`
- template<class `_CharT2`, class `_Alloc2`>
`rope<_CharT2, _Alloc2>` **operator+** (`const rope<_CharT2, _Alloc2>` &__left, `const rope<_CharT2, _Alloc2>` &__right)
- template<class `_CharT2`, class `_Alloc2`>
`rope<_CharT2, _Alloc2>` **operator+** (`const rope<_CharT2, _Alloc2>` &__left, `const _CharT2 *` __right)
- template<class `_CharT2`, class `_Alloc2`>
`rope<_CharT2, _Alloc2>` **operator+** (`const rope<_CharT2, _Alloc2>` &__left, `_CharT2` __right)

4.62.1 Detailed Description

```
template<class _CharT, class _Alloc> class __gnu_cxx::rope<_CharT, _Alloc>
```

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Definition at line 1521 of file rope.

The documentation for this class was generated from the following files:

- [rope](#)
- [ropeimpl.h](#)

4.63 `__gnu_cxx::select1st<_Pair>` Struct Template Reference

Inherits `std::_Select1st<_Pair>`.

Public Types

- typedef `_Pair` [argument_type](#)
- typedef `_Pair::first_type` [result_type](#)

Public Member Functions

- `_Pair::first_type & operator() (_Pair &__x) const`
- `const _Pair::first_type & operator() (const _Pair &__x) const`
- `template<typename _Pair2> _Pair2::first_type & operator() (_Pair2 &__x) const`
- `template<typename _Pair2> const _Pair2::first_type & operator() (const _Pair2 &__x) const`

4.63.1 Detailed Description

```
template<class _Pair> struct __gnu_cxx::select1st<_Pair>
```

An [SGI extension](#) .

Definition at line 200 of file ext/functional.

4.63.2 Member Typedef Documentation

4.63.2.1 `typedef _Pair std::unary_function<_Pair, _Pair::first_type>::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.63.2.2 `typedef _Pair::first_type std::unary_function<_Pair, _Pair::first_type>::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.64 `__gnu_cxx::select2nd<_Pair>` Struct Template Reference

Inherits `std::_Select2nd<_Pair>`.

Public Types

- `typedef _Pair` [argument_type](#)
- `typedef _Pair::second_type` [result_type](#)

Public Member Functions

- `_Pair::second_type & operator() (_Pair &__x) const`
- `const _Pair::second_type & operator() (const _Pair &__x) const`

4.64.1 Detailed Description

`template<class _Pair>struct __gnu_cxx::select2nd<_Pair>`

An [SGI extension](#).

Definition at line 205 of file `ext/functional`.

4.64.2 Member Typedef Documentation

4.64.2.1 `typedef _Pair std::unary_function<_Pair, _Pair::second_type>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.64.2.2 `typedef _Pair::second_type std::unary_function<_Pair, _Pair::second_type>::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.65 `__gnu_cxx::slist<_Tp, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::_Slist_base<_Tp, _Alloc>`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Slist_iterator< _Tp, const _Tp &, const _Tp * > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Slist_iterator< _Tp, _Tp &, _Tp * > iterator`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `slist (const allocator_type &__a=allocator_type())`
- `slist (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())`
- `slist (size_type __n)`
- `template<class _InputIterator >
slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `slist (const slist &__x)`
- `template<class _Integer >
void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<class _InputIterator >
void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_fill_assign (size_type __n, const _Tp &__val)`
- `void assign (size_type __n, const _Tp &__val)`
- `template<class _InputIterator >
void assign (_InputIterator __first, _InputIterator __last)`
- `iterator before_begin ()`
- `const_iterator before_begin () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `void clear ()`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __pos)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase_after (iterator __pos)`
- `iterator erase_after (iterator __before_first, iterator __last)`
- `reference front ()`
- `const_reference front () const`
- `allocator_type get_allocator () const`
- `iterator insert (iterator __pos, const value_type &__x)`
- `iterator insert (iterator __pos)`
- `void insert (iterator __pos, size_type __n, const value_type &__x)`
- `template<class _InIterator >
void insert (iterator __pos, _InIterator __first, _InIterator __last)`

- iterator **insert_after** (iterator __pos, const value_type &__x)
- iterator **insert_after** (iterator __pos)
- void **insert_after** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
void **insert_after** (iterator __pos, _InIterator __first, _InIterator __last)
- size_type **max_size** () const
- void **merge** (slist &__x)
- template<class _StrictWeakOrdering >
void **merge** (slist &, _StrictWeakOrdering)
- slist & **operator=** (const slist &__x)
- void **pop_front** ()
- iterator **previous** (const_iterator __pos)
- const_iterator **previous** (const_iterator __pos) const
- void **push_front** (const value_type &__x)
- void **push_front** ()
- void **remove** (const _Tp &__val)
- template<class _Predicate >
void **remove_if** (_Predicate __pred)
- void **resize** (size_type new_size, const _Tp &__x)
- void **resize** (size_type new_size)
- void **reverse** ()
- size_type **size** () const
- void **sort** ()
- template<class _StrictWeakOrdering >
void **sort** (_StrictWeakOrdering __comp)
- void **splice** (iterator __pos, slist &__x)
- void **splice** (iterator __pos, slist &__x, iterator __i)
- void **splice** (iterator __pos, slist &__x, iterator __first, iterator __last)
- void **splice_after** (iterator __pos, iterator __before_first, iterator __before_last)
- void **splice_after** (iterator __pos, iterator __prev)
- void **splice_after** (iterator __pos, slist &__x)
- void **swap** (slist &__x)
- void **unique** ()
- template<class _BinaryPredicate >
void **unique** (_BinaryPredicate __pred)

Private Types

- typedef _Alloc::template
rebind<_Slist_node<_Tp>
>::other **_Node_alloc**

Private Member Functions

- _Slist_node_base * **_M_erase_after** (_Slist_node_base * __pos)
- _Slist_node_base * **_M_erase_after** (_Slist_node_base *, _Slist_node_base *)
- _Slist_node<_Tp> * **_M_get_node** ()
- void **_M_put_node** (_Slist_node<_Tp> * __p)

Private Attributes

- `_Slist_node_base _M_head`

4.65.1 Detailed Description

```
template<class _Tp, class _Alloc = allocator<_Tp>>class __gnu_cxx::slist<_Tp, _Alloc>
```

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

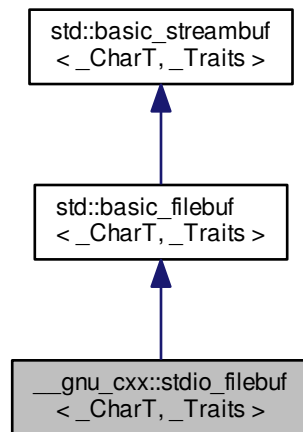
Definition at line 292 of file `slist`.

The documentation for this class was generated from the following file:

- [slist](#)

4.66 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



Public Types

- `typedef codecvt< char_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic_filebuf < char_type, traits_type > __filebuf_type`

- `typedef traits_type::state_type __state_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef std::size_t size_t`
- `typedef _Traits traits_type`

Public Member Functions

- `stdio_filebuf ()`
- `stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
- `stdio_filebuf (std::__c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
- `virtual ~stdio_filebuf ()`
- `__filebuf_type * close ()`
- `int fd ()`
- `std::__c_file * file ()`
- `locale getloc () const`
- `streamsize in_avail ()`
- `bool is_open () const throw ()`
- `__filebuf_type * open (const char * __s, ios_base::openmode __mode)`
- `__filebuf_type * open (const std::string & __s, ios_base::openmode __mode)`
- `locale pubimbue (const locale & __loc)`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type * __s, streamsize __n)`
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- `streamsize sputn (const char_type * __s, streamsize __n)`
- `int_type sungetc ()`
- `basic_streambuf * pubsetbuf (char_type * __s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `int pubsync ()`

Protected Member Functions

- `void __safe_gbump (streamsize __n)`
- `void __safe_pbump (streamsize __n)`
- `void _M_allocate_internal_buffer ()`
- `bool _M_convert_to_external (char_type *, streamsize)`
- `void _M_create_pback ()`
- `void _M_destroy_internal_buffer () throw ()`
- `void _M_destroy_pback () throw ()`
- `int _M_get_ext_pos (__state_type & __state)`

- `pos_type _M_seek` (`off_type __off`, `ios_base::seekdir __way`, `__state_type __state`)
 - `void _M_set_buffer` (`streamsize __off`)
 - `bool _M_terminate_output` ()
 - `void gbump` (`int __n`)
 - `virtual void imbue` (`const locale &__loc`)
 - `virtual int_type overflow` (`int_type __c=_Traits::eof()`)
 - `virtual int_type pbackfail` (`int_type __c=_Traits::eof()`)
 - `void pbump` (`int __n`)
 - `virtual pos_type seekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
 - `virtual pos_type seekpos` (`pos_type __pos`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
 - `virtual __streambuf_type * setbuf` (`char_type *__s`, `streamsize __n`)
 - `void setg` (`char_type *__gbeg`, `char_type *__gnext`, `char_type *__gend`)
 - `void setp` (`char_type *__pbeg`, `char_type *__pend`)
 - `virtual streamsize showmanyc` ()
 - `virtual int sync` ()
 - `virtual int_type uflow` ()
 - `virtual int_type underflow` ()
 - `virtual streamsize xsgetn` (`char_type *__s`, `streamsize __n`)
 - `virtual streamsize xsputn` (`const char_type *__s`, `streamsize __n`)
-
- `char_type * eback` () `const`
 - `char_type * gptr` () `const`
 - `char_type * egptr` () `const`
-
- `char_type * pbase` () `const`
 - `char_type * pptr` () `const`
 - `char_type * epptr` () `const`

Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `locale _M_buf_locale`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `bool _M_reading`

- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

4.66.1 Detailed Description

`template<typename _CharT, typename _Traits = std::char_traits<_CharT>> class __gnu_cxx::stdio_filebuf< _CharT, _Traits >`

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 50 of file `stdio_filebuf.h`.

4.66.2 Constructor & Destructor Documentation

4.66.2.1 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf ()` [inline]

deferred initialization

Definition at line 65 of file `stdio_filebuf.h`.

4.66.2.2 `template<typename _CharT, typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size = static_cast<size_t>(BUFSIZ))`

Parameters

<code>__fd</code>	An open file descriptor.
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 128 of file `stdio_filebuf.h`.

4.66.2.3 `template<typename _CharT, typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (std::_c_file * __f, std::ios_base::openmode __mode, size_t __size = static_cast<size_t>(BUFSIZ))`

Parameters

<code>__f</code>	An open FILE*.
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars. Defaults to system's BUFSIZ.

This constructor associates a file stream buffer with an open C FILE*. The FILE* will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 144 of file `stdio_filebuf.h`.

4.66.2.4 `template<typename _CharT, typename _Traits> __gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf ()`
`[virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 123 of file `stdio_filebuf.h`.

4.66.3 Member Function Documentation

4.66.3.1 `template<typename _CharT, typename _Traits> void std::basic_filebuf<_CharT, _Traits>::_M_create_pback ()`
`[inline], [protected], [inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 177 of file `fstream`.

4.66.3.2 `template<typename _CharT, typename _Traits> void std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback ()`
`throw () [inline], [protected], [inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 194 of file `fstream`.

4.66.3.3 `template<typename _CharT, typename _Traits> void std::basic_filebuf<_CharT, _Traits>::_M_set_buffer (`
`streamsize __off) [inline], [protected], [inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `eptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 395 of file `fstream`.

4.66.3.4 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::~__filebuf_type *`
`std::basic_filebuf<_CharT, _Traits>::close () [inherited]`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file `fstream.tcc`.

Referenced by `std::basic_ifstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_fstream<_CharT, _Traits>::close()`, and `std::basic_filebuf<char_type, traits_type>::~~basic_filebuf()`.

4.66.3.5 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback () const`
`[inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 482 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, and `std::basic_streambuf<char_type, traits_type>::sungetc()`.

4.66.3.6 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr () const`
`[inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 488 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.66.3.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eptr () const`
`[inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 535 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputc()`.

4.66.3.8 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf<_CharT, _Traits>::fd () [inline]`

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 109 of file `stdio_filebuf.h`.

4.66.3.9 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> std::_c_file* __gnu_cxx::stdio_filebuf<_CharT, _Traits>::file () [inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 119 of file `stdio_filebuf.h`.

4.66.3.10 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::gbump (int __n) [inline], [protected], [inherited]`

Moving the read position.

Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 498 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, and `std::basic_streambuf<char_type, traits_type>::uflow()`.

4.66.3.11 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::getloc () const [inline], [inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

4.66.3.12 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 485 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, and `std::basic_streambuf<char_type, traits_type>::uflow()`.

4.66.3.13 `template<typename _CharT, typename _Traits> void std::basic_filebuf<_CharT, _Traits>::imbue (const locale & __loc)` `[protected]`, `[virtual]`, `[inherited]`

Changes translations.

Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 910 of file `fstream.tcc`.

References `std::ios_base::cur`.

4.66.3.14 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::in_avail ()` `[inline]`, `[inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file `streambuf`.

4.66.3.15 `template<typename _CharT, typename _Traits> bool std::basic_filebuf<_CharT, _Traits>::is_open () const throw ()` `[inline]`, `[inherited]`

Returns true if the external file is open.

Definition at line 227 of file `fstream`.

Referenced by `std::basic_ifstream<_CharT, _Traits>::is_open()`, `std::basic_ofstream<_CharT, _Traits>::is_open()`, and `std::basic_fstream<_CharT, _Traits>::is_open()`.

4.66.3.16 `template<typename _CharT, typename _Traits> basic_filebuf< _CharT, _Traits >::__filebuf_type *
std::basic_filebuf< _CharT, _Traits >::open (const char * __s, ios_base::openmode __mode)
[inherited]`

Opens an external file.

Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +
+ | ios_base Flag combination stdio equivalent | | binary in out trunc app | +
+ | + w | | + + a | | + a | | + + w | | + r | | + + r+ | | + + + w+ | | + + + a+ | | + + a+ | +
+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + + w+b | | + + + + a+b | | +
+ + a+b | +

Definition at line 94 of file fstream.tcc.

References `std::ios_base::ate`, `std::ios_base::end`, and `std::basic_filebuf< _CharT, _Traits >::open()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, and `std::basic_fstream< _CharT, _Traits >::open()`.

4.66.3.17 `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_filebuf< _CharT, _Traits >::open (const std::string & __s, ios_base::openmode __mode) [inline], [inherited]`

Opens an external file.

Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Returns

`this` on success, NULL on failure

Definition at line 280 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

4.66.3.18 `template<typename _CharT, typename _Traits> basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (int_type __c = _Traits::eof()) [protected], [virtual], [inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 422 of file `fstream.tcc`.

References `std::ios_base::cur`, and `std::ios_base::out`.

4.66.3.19 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::pbackfail (int_type __c = _Traits::eof())` `[protected]`, `[virtual]`, `[inherited]`

Tries to back up the input sequence.

Parameters

<code>__c</code>	The character to be inserted back into the sequence.
------------------	--

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 363 of file `fstream.tcc`.

References `std::ios_base::cur`, and `std::ios_base::in`.

4.66.3.20 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::pbase ()` `const` `[inline]`, `[protected]`, `[inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 529 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.66.3.21 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::pbump (int __n)`
`[inline], [protected], [inherited]`

Moving the write position.

Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the write position without returning any data.

Definition at line 545 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::putc()`.

4.66.3.22 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::pptr () const`
`[inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 532 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::putc()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.66.3.23 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::pubimbue (const locale & __loc)`
`[inline], [inherited]`

Entry point for `imbue()`.

Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

Returns

The previous locale.

Calls the derived `imbue(__loc)`.

Definition at line 209 of file `streambuf`.

4.66.3.24 `template<typename _CharT, typename _Traits> pos_type std::basic_streambuf<_CharT, _Traits>::pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]`

Alters the stream position.

Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekoff` function.

Definition at line 251 of file `streambuf`.

4.66.3.25 `template<typename _CharT, typename _Traits> pos_type std::basic_streambuf<_CharT, _Traits>::pubseekpos (pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]`

Alters the stream position.

Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 263 of file `streambuf`.

4.66.3.26 `template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf<_CharT, _Traits>::pubsetbuf (char_type* __s, streamsize __n) [inline], [inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file `streambuf`.

4.66.3.27 `template<typename _CharT, typename _Traits> int std::basic_streambuf<_CharT, _Traits>::pubsync () [inline], [inherited]`

Calls virtual `sync` function.

Definition at line 271 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

4.66.3.28 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sbumpc () [inline], [inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::istreambuf_iterator<_CharT, _Traits>::operator++()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

4.66.3.29 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::pos_type
std::basic_filebuf<_CharT, _Traits>::seekoff (off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [protected], [virtual], [inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 713 of file `fstream.tcc`.

References `std::ios_base::cur`.

4.66.3.30 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<
_CharT, _Traits>::seekpos (pos_type, ios_base::openmode = ios_base::in | ios_base::out)
[protected], [virtual], [inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 773 of file `fstream.tcc`.

References `std::ios_base::beg`.

4.66.3.31 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::__streambuf_type *
std::basic_filebuf<_CharT, _Traits>::setbuf (char_type * __s, streamsize __n) [protected],
[virtual], [inherited]`

Manipulates the buffer.

Parameters

<code>__s</code>	Pointer to a buffer area.
<code>__n</code>	Size of <code>__s</code> .

Returns

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.-html> for more.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 684 of file `fstream.tcc`.

4.66.3.32 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)` `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 509 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, and `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.66.3.33 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::setp (char_type * __pbeg, char_type * __pend)` `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 555 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.66.3.34 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sgetc ()` `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

4.66.3.35 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (char_type * __s, streamsize __n)` `[inline]`, `[inherited]`

Entry point for `xsgetn`.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 357 of file `streambuf`.

4.66.3.36 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf<_CharT, _Traits>::showmanyc ()` `[protected]`, `[virtual]`, `[inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 178 of file `fstream.tcc`.

References `std::ios_base::binary`, and `std::ios_base::in`.

4.66.3.37 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::snextc ()` `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sputc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

4.66.3.38 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sputback (char_type __c) [inline], [inherited]`

Pushing characters back into the input stream.

Parameters

<code>__c</code>	The character to push back.
------------------	-----------------------------

Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 372 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

4.66.3.39 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sputc (char_type __c) [inline], [inherited]`

Entry point for all single-character output functions.

Parameters

<code>__c</code>	A character to output.
------------------	------------------------

Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(-__c)`.

Definition at line 424 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

4.66.3.40 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::sputn (const char_type * __s, streamsize __n) [inline], [inherited]`

Entry point for all single-character output functions.

Parameters

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 450 of file `streambuf`.

4.66.3.41 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sungetc ()`
`[inline], [inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 397 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

4.66.3.42 `template<typename _CharT, typename _Traits> int std::basic_filebuf<_CharT, _Traits>::sync ()`
`[protected], [virtual], [inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 893 of file `fstream.tcc`.

4.66.3.43 `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf<_CharT, _Traits>::uflow ()`
`[inline], [protected], [virtual], [inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 700 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sbumpc()`.

4.66.3.44 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::underflow()` `[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 204 of file `fstream.tcc`.

References `std::ios_base::in`, and `std::min()`.

4.66.3.45 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf<_CharT, _Traits>::xsgetn(char_type * __s, streamsize __n)` `[protected]`, `[virtual]`, `[inherited]`

Multiple character extraction.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 549 of file `fstream.tcc`.

References `std::ios_base::in`.

4.66.3.46 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf<_CharT, _Traits>::xsputn(const char_type * __s, streamsize __n)` `[protected]`, `[virtual]`, `[inherited]`

Multiple character insertion.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 637 of file `fstream.tcc`.

References `std::min()`, and `std::ios_base::out`.

4.66.4 Member Data Documentation

4.66.4.1 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf`
[protected], [inherited]

Pointer to the beginning of internal buffer.

Definition at line 114 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, and `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.66.4.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale`
[protected], [inherited]

Current locale setting.

Definition at line 192 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`, `std::basic_streambuf<char_type, traits_type>::getloc()`, and `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

4.66.4.3 `template<typename _CharT, typename _Traits> size_t std::basic_filebuf<_CharT, _Traits>::_M_buf_size`
[protected], [inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 121 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.66.4.4 `template<typename _CharT, typename _Traits> char* std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`
[protected], [inherited]

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 156 of file `fstream`.

4.66.4.5 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`
[protected], [inherited]

Size of buffer held by `_M_ext_buf`.

Definition at line 161 of file `fstream`.

4.66.4.6 `template<typename _CharT, typename _Traits> const char* std::basic_filebuf<_CharT, _Traits>::_M_ext_next`
[protected], [inherited]

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 168 of file `fstream`.

4.66.4.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg`
[protected], [inherited]

Start of get area.

Definition at line 184 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::eback()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

4.66.4.8 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur`
[protected], [inherited]

Current read area.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::gbump()`, `std::basic_streambuf<char_type, traits_type>::gptr()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

4.66.4.9 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_end`
[protected], [inherited]

End of get area.

Definition at line 186 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::egptr()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

4.66.4.10 `template<typename _CharT, typename _Traits> ios_base::openmode std::basic_filebuf<_CharT, _Traits>::_M_mode`
[protected], [inherited]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 99 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.66.4.11 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg`
[protected], [inherited]

Start of put area.

Definition at line 187 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pbase()`, and `std::basic_streambuf<char_type, traits_type>::setp()`.

`type >::setp()`.

4.66.4.12 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::M_out_cur`
`[protected], [inherited]`

Current put area.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pbump()`, `std::basic_streambuf<char_type, traits_type>::pptr()`, and `std::basic_streambuf<char_type, traits_type>::setp()`.

4.66.4.13 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::M_out_end`
`[protected], [inherited]`

End of put area.

Definition at line 189 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::epptr()`, and `std::basic_streambuf<char_type, traits_type>::setp()`.

4.66.4.14 `template<typename _CharT, typename _Traits> char_type std::basic_filebuf<_CharT, _Traits>::M_pback`
`[protected], [inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 142 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::M_create_pback()`.

4.66.4.15 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::M_pback_cur_save`
`[protected], [inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 143 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::M_create_pback()`, and `std::basic_filebuf<char_type, traits_type>::M_destroy_pback()`.

4.66.4.16 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::M_pback_end_save`
`[protected], [inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 144 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::M_create_pback()`, and `std::basic_filebuf<char_type, traits_type>::M_destroy_pback()`.

4.66.4.17 `template<typename _CharT, typename _Traits> bool std::basic_filebuf<_CharT, _Traits>::_M_pback_init`
`[protected], [inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 145 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, and `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`.

4.66.4.18 `template<typename _CharT, typename _Traits> bool std::basic_filebuf<_CharT, _Traits>::_M_reading`
`[protected], [inherited]`

`_M_reading == false` && `_M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true` && `_M_writing == true` is unused.

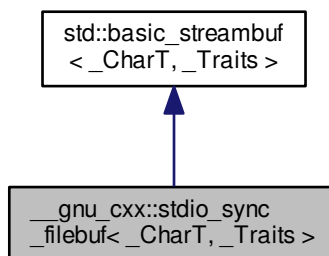
Definition at line 133 of file `fstream`.

The documentation for this class was generated from the following file:

- [stdio_filebuf.h](#)

4.67 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`:



Public Types

- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`

- typedef `_Traits` **traits_type**
- typedef `basic_streambuf`
`< char_type, traits_type > __streambuf_type`

Public Member Functions

- **stdio_sync_filebuf** (`std::__c_file *__f`)
- `std::__c_file *const file` ()
- locale `getloc` () const
- streamsize `in_avail` ()
- locale `pubimbue` (const locale &__loc)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- streamsize `sgetn` (`char_type *__s`, streamsize __n)
- `int_type snextc` ()
- `int_type sputbackc` (`char_type __c`)
- `int_type sputc` (`char_type __c`)
- streamsize `sputn` (const `char_type *__s`, streamsize __n)
- `int_type sungetc` ()
- `basic_streambuf * pubsetbuf` (`char_type *__s`, streamsize __n)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()

Protected Member Functions

- void `__safe_gbump` (streamsize __n)
- void `__safe_pbump` (streamsize __n)
- void `gbump` (int __n)
- virtual void `imbue` (const locale &__loc)
- virtual `int_type overflow` (`int_type __c=traits_type::eof()`)
- virtual `int_type pbackfail` (`int_type __c=traits_type::eof()`)
- void `pbump` (int __n)
- virtual `std::streampos seekoff` (`std::streamoff __off`, `std::ios_base::seekdir __dir`, `std::ios_base::openmode=std::ios_base::in|std::ios_base::out`)
- virtual `pos_type seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode=ios_base::in|ios_base::out`)
- virtual `std::streampos seekpos` (`std::streampos __pos`, `std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out`)
- virtual `pos_type seekpos` (`pos_type`, `ios_base::openmode=ios_base::in|ios_base::out`)
- virtual `basic_streambuf`
`< char_type, _Traits > * setbuf` (`char_type *`, streamsize)
- void `setg` (`char_type *__gbeg`, `char_type *__gnext`, `char_type *__gend`)
- void `setp` (`char_type *__pbeg`, `char_type *__pend`)
- virtual streamsize `showmanyc` ()
- virtual `int sync` ()
- `int_type syncgetc` ()

- `template<>`
`stdio_sync_filebuf< char >`
`::int_type syncgetc ()`
- `template<>`
`stdio_sync_filebuf< wchar_t >`
`::int_type syncgetc ()`
- `int_type syncputc (int_type __c)`
- `template<>`
`stdio_sync_filebuf< char >`
`::int_type syncputc (int_type __c)`
- `template<>`
`stdio_sync_filebuf< wchar_t >`
`::int_type syncputc (int_type __c)`
- `int_type syncungetc (int_type __c)`
- `template<>`
`stdio_sync_filebuf< char >`
`::int_type syncungetc (int_type __c)`
- `template<>`
`stdio_sync_filebuf< wchar_t >`
`::int_type syncungetc (int_type __c)`
- virtual `int_type uflow ()`
- virtual `int_type underflow ()`
- virtual `std::streamsize xsgetn (char_type *__s, std::streamsize __n)`
- `template<>`
`std::streamsize xsgetn (char *__s, std::streamsize __n)`
- `template<>`
`std::streamsize xsgetn (wchar_t *__s, std::streamsize __n)`
- virtual `streamsize xsgetn (char_type *__s, streamsize __n)`
- virtual `std::streamsize xspu (const char_type *__s, std::streamsize __n)`
- `template<>`
`std::streamsize xspu (const char *__s, std::streamsize __n)`
- `template<>`
`std::streamsize xspu (const wchar_t *__s, std::streamsize __n)`
- virtual `streamsize xspu (const char_type *__s, streamsize __n)`
- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * ep_ptr () const`

Protected Attributes

- `locale _M_buf_locale`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`

4.67.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>> class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 56 of file `stdio_sync_filebuf.h`.

4.67.2 Member Typedef Documentation

```
4.67.2.1 template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type>
        std::basic_streambuf<_CharT, _Traits>::__streambuf_type [inherited]
```

This is a non-standard type.

Definition at line 138 of file `streambuf`.

4.67.3 Member Function Documentation

```
4.67.3.1 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback ( )
        const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 482 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, and `std::basic_streambuf<char_type, traits_type>::sungetc()`.

```
4.67.3.2 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr ( )
        const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 488 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.67.3.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eptr ()`
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 535 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputc()`.

4.67.3.4 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> std::__c_file* const`
`__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file () [inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 88 of file `stdio_sync_filebuf.h`.

4.67.3.5 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::gbump (int __n)`
`[inline], [protected], [inherited]`

Moving the read position.

Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 498 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, and `std::basic_streambuf<char_type, traits_type>::uflow()`.

4.67.3.6 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::getloc () const`
`[inline], [inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

4.67.3.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::gptr ()`
`const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 485 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, and `std::basic_streambuf<char_type, traits_type>::uflow()`.

4.67.3.8 `template<typename _CharT, typename _Traits> virtual void std::basic_streambuf<_CharT, _Traits>::imbue (const locale & __loc)` `[inline], [protected], [virtual], [inherited]`

Changes translations.

Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 576 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

4.67.3.9 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::in_avail ()`
`[inline], [inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file `streambuf`.

4.67.3.10 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type
 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::overflow (int_type __c = traits_type::eof())
 [inline], [protected], [virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 141 of file `stdio_sync_filebuf.h`.

4.67.3.11 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type
 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pbackfail (int_type __c = traits_type::eof())
 [inline], [protected], [virtual]`

Tries to back up the input sequence.

Parameters

<code>__c</code>	The character to be inserted back into the sequence.
------------------	--

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 116 of file `stdio_sync_filebuf.h`.

4.67.3.12 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::pbase ()`
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `pptr()` returns the end pointer for the output sequence

Definition at line 529 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.67.3.13 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::pbump (int __n)`
`[inline], [protected], [inherited]`

Moving the write position.

Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the write position without returning any data.

Definition at line 545 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputc()`.

4.67.3.14 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::pptr ()`
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `pptr()` returns the end pointer for the output sequence

Definition at line 532 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputc()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.67.3.15 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::pubimbue (const locale & __loc)`
`[inline], [inherited]`

Entry point for `imbue()`.

Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

Returns

The previous locale.

Calls the derived `imbue(__loc)`.

Definition at line 209 of file `streambuf`.

```
4.67.3.16 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf<_CharT, _Traits>::pubseekoff (
    off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out )
    [inline], [inherited]
```

Alters the stream position.

Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekoff` function.

Definition at line 251 of file `streambuf`.

```
4.67.3.17 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf<_CharT, _Traits>::pubseekpos
    ( pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline],
    [inherited]
```

Alters the stream position.

Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 263 of file `streambuf`.

```
4.67.3.18 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf<_CharT, _Traits>
    >::pubsetbuf ( char_type * __s, streamsize __n ) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file `streambuf`.

```
4.67.3.19 template<typename _CharT, typename _Traits> int std::basic_streambuf<_CharT, _Traits>::pubsync ( )
    [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 271 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

4.67.3.20 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sbumpc ()`
`[inline], [inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_streambuf_iterator<_CharT, _Traits>::operator++()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

4.67.3.21 `template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekoff (off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual], [inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 602 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekoff()`.

4.67.3.22 `template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekpos (pos_type, ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual], [inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 614 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekpos()`.

4.67.3.23 `template<typename _CharT, typename _Traits> virtual basic_streambuf<char_type, _Traits>*
std::basic_streambuf<_CharT, _Traits>::setbuf (char_type *, streamsize) [inline], [protected], [virtual], [inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 591 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubsetbuf()`.

4.67.3.24 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)` `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 509 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, and `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.67.3.25 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::setp (char_type * __pbeg, char_type * __pend)` `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 555 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`.

4.67.3.26 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sgetc ()` `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

4.67.3.27 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (char_type * __s, streamsize __n) [inline], [inherited]`

Entry point for `xsgetn`.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 357 of file `streambuf`.

4.67.3.28 `template<typename _CharT, typename _Traits> virtual streamsize std::basic_streambuf<_CharT, _Traits>::showmanyc () [inline], [protected], [virtual], [inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 649 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::in_avail()`.

4.67.3.29 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sputc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

4.67.3.30 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (char_type __c) [inline], [inherited]`

Pushing characters back into the input stream.

Parameters

<code>__c</code>	The character to push back.
------------------	-----------------------------

Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 372 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

4.67.3.31 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sputc (char_type __c) [inline], [inherited]`

Entry point for all single-character output functions.

Parameters

<code>__c</code>	A character to output.
------------------	------------------------

Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(-__c)`.

Definition at line 424 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

4.67.3.32 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::sputn (const char_type * __s, streamsize __n) [inline], [inherited]`

Entry point for all single-character output functions.

Parameters

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 450 of file `streambuf`.

4.67.3.33 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sungetc ()`
`[inline], [inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 397 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

4.67.3.34 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::sync (void)`
`[inline], [protected], [virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 160 of file `stdio_sync_filebuf.h`.

4.67.3.35 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow ()`
`[inline], [protected], [virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 108 of file `stdio_sync_filebuf.h`.

4.67.336 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow () [inline], [protected],
[virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 101 of file `stdio_sync_filebuf.h`.

4.67.337 `template<typename _CharT, typename _Traits> streamsize basic_streambuf<_CharT, _Traits>::xsgetn (
char_type * __s, streamsize __n) [protected], [virtual], [inherited]`

Multiple character extraction.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic_filebuf<_CharT, _Traits>](#), [std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>](#), and [std::basic_filebuf<char_type, traits_type>](#).

Definition at line 46 of file `streambuf.tcc`.

References `std::min()`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sgetn()`.

4.67.338 `template<typename _CharT, typename _Traits> streamsize basic_streambuf<_CharT, _Traits>::xsputn (const
char_type * __s, streamsize __n) [protected], [virtual], [inherited]`

Multiple character insertion.

Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 80 of file `streambuf.tcc`.

References `std::min()`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputn()`.

4.67.4 Member Data Documentation

4.67.4.1 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale`
[protected], [inherited]

Current locale setting.

Definition at line 192 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`, `std::basic_streambuf<char_type, traits_type>::getloc()`, and `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

4.67.4.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg`
[protected], [inherited]

Start of get area.

Definition at line 184 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::eback()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

4.67.4.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur`
[protected], [inherited]

Current read area.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::gbump()`, `std::basic_streambuf<char_type, traits_type>::gptr()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

4.67.4.4 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_end`
[protected], [inherited]

End of get area.

Definition at line 186 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

4.67.4.5 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_out_beg`
`[protected], [inherited]`

Start of put area.

Definition at line 187 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbase()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

4.67.4.6 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_out_cur`
`[protected], [inherited]`

Current put area.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbump()`, `std::basic_streambuf< char_type, traits_type >::pptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

4.67.4.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_out_end`
`[protected], [inherited]`

End of put area.

Definition at line 189 of file `streambuf`.

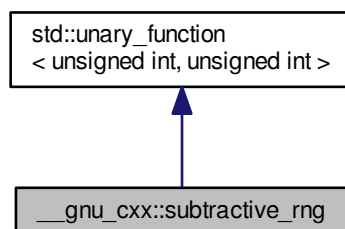
Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

The documentation for this class was generated from the following file:

- [stdio_sync_filebuf.h](#)

4.68 __gnu_cxx::subtracive_rng Class Reference

Inheritance diagram for `__gnu_cxx::subtracive_rng`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- [subtractive_rng](#) (unsigned int `__seed`)
- [subtractive_rng](#) ()
- void `_M_initialize` (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

4.68.1 Detailed Description

The `subtractive_rng` class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits. Definition at line 352 of file `ext/functional`.

4.68.2 Member Typedef Documentation

4.68.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`
[[inherited](#)]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.68.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`
[[inherited](#)]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

4.68.3 Constructor & Destructor Documentation

4.68.3.1 `__gnu_cxx::subtractive_rng::subtractive_rng (unsigned int __seed)` [[inline](#)]

Ctor allowing you to initialize the seed.

Definition at line 394 of file `ext/functional`.

4.68.3.2 `__gnu_cxx::subtractive_rng::subtractive_rng ()` [[inline](#)]

Default ctor; initializes its state with some number you don't see.

Definition at line 398 of file `ext/functional`.

4.68.4 Member Function Documentation

4.68.4.1 `unsigned int __gnu_cxx::subtractive_rng::operator() (unsigned int __limit)` [[inline](#)]

Returns a number less than the argument.

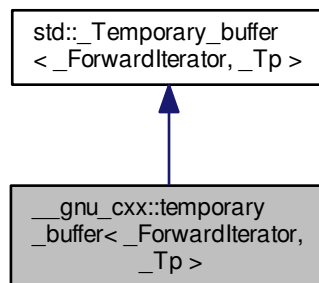
Definition at line 363 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

4.69 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



Public Types

- typedef pointer **iterator**
- typedef `value_type * pointer`
- typedef `ptrdiff_t size_type`
- typedef `_Tp value_type`

Public Member Functions

- [temporary_buffer](#) (`_ForwardIterator __first, _ForwardIterator __last`)
- [~temporary_buffer](#) ()
- iterator [begin](#) ()
- iterator [end](#) ()
- `size_type requested_size` () const
- `size_type size` () const

Protected Attributes

- pointer **_M_buffer**
- `size_type _M_len`
- `size_type _M_original_len`

4.69.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>struct __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>
```

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

Definition at line 183 of file `ext/memory`.

4.69.2 Constructor & Destructor Documentation

```
4.69.2.1 template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
        __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>::temporary_buffer ( _ForwardIterator __first,
        _ForwardIterator __last ) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

Definition at line 186 of file `ext/memory`.

```
4.69.2.2 template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
        __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>::~temporary_buffer ( ) [inline]
```

Destroys objects and frees storage.

Definition at line 190 of file `ext/memory`.

4.69.3 Member Function Documentation

```
4.69.3.1 template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer<_ForwardIterator, _Tp>::begin ( ) [inline], [inherited]
```

As per Table mumble.

Definition at line 151 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

```
4.69.3.2 template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer<_ForwardIterator, _Tp>::end ( ) [inline], [inherited]
```

As per Table mumble.

Definition at line 156 of file `stl_tempbuf.h`.

```
4.69.3.3 template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size ( ) const [inline], [inherited]
```

Returns the size requested by the constructor; may be `>size()`.

Definition at line 146 of file stl_tempbuf.h.

Referenced by std::stable_partition().

4.69.3.4 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const` `[inline]`, `[inherited]`

As per Table mumble.

Definition at line 141 of file stl_tempbuf.h.

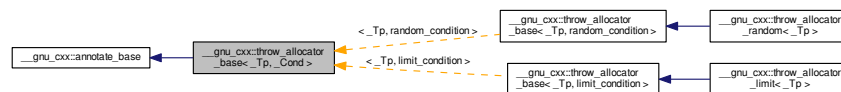
Referenced by std::inplace_merge(), std::stable_partition(), and std::stable_sort().

The documentation for this struct was generated from the following file:

- [ext/memory](#)

4.70 __gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference

Inheritance diagram for __gnu_cxx::throw_allocator_base< _Tp, _Cond >:



Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `pointer address (reference __x) const noexcept`
- `const_pointer address (const_reference __x) const noexcept`
- `pointer allocate (size_type __n, std::allocator< void >::const_pointer hint=0)`
- `void check_allocated (void *p, size_t size)`
- `void check_allocated (pointer __p, size_type __n)`
- `void check_allocated (size_type __n)`
- `template<typename _Up, typename... _Args>`
`void construct (_Up *__p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type __n)`
- `template<typename _Up >`
`void destroy (_Up *__p)`

- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)

4.70.1 Detailed Description

template<typename _Tp, typename _Cond>class __gnu_cxx::throw_allocator_base< _Tp, _Cond >

Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templated code.

Note: Deallocate not allowed to throw.

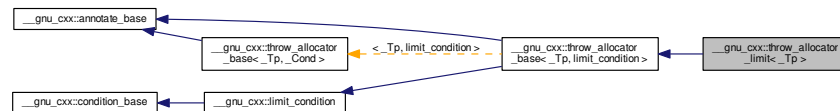
Definition at line 634 of file throw_allocator.h.

The documentation for this class was generated from the following file:

- [throw_allocator.h](#)

4.71 __gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference

Inheritance diagram for __gnu_cxx::throw_allocator_limit< _Tp >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef [std::true_type](#) **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_limit** (const [throw_allocator_limit](#) &) noexcept
- template<typename _Tp1 >
 throw_allocator_limit (const [throw_allocator_limit](#)< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (size_type __n)
- void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

4.71.1 Detailed Description

template<typename _Tp>struct __gnu_cxx::throw_allocator_limit< _Tp >

Allocator throwing via limit condition.

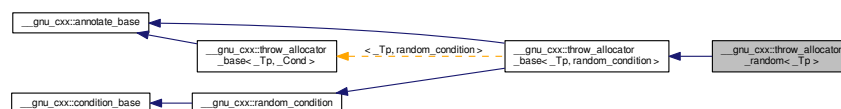
Definition at line 737 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.72 __gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference

Inheritance diagram for __gnu_cxx::throw_allocator_random< _Tp >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef [std::true_type](#) **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_random** (const [throw_allocator_random](#) &) noexcept
- template<typename _Tp1 >
 throw_allocator_random (const [throw_allocator_random](#)<_Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (size_type __n)
- void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)
- size_type **max_size** () const noexcept
- void **seed** (unsigned long __s)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

4.72.1 Detailed Description

template<typename _Tp>struct `__gnu_cxx::throw_allocator_random<_Tp>`

Allocator throwing via random condition.

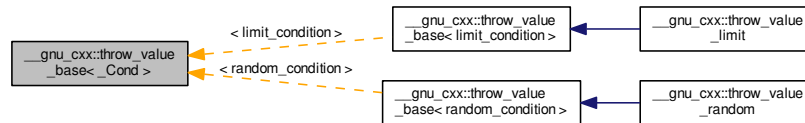
Definition at line 758 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.73 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_value_base<_Cond>`:



Public Types

- typedef `_Cond` **condition_type**

Public Member Functions

- **throw_value_base** (const `throw_value_base` &__v)
- **throw_value_base** (`throw_value_base` &&)=default
- **throw_value_base** (const std::size_t __i)
- `throw_value_base` & **operator++** ()
- `throw_value_base` & **operator=** (const `throw_value_base` &__v)
- `throw_value_base` & **operator=** (`throw_value_base` &&)=default

Public Attributes

- std::size_t **M_i**

4.73.1 Detailed Description

```
template<typename _Cond>struct __gnu_cxx::throw_value_base<_Cond>
```

Class with exception generation control. Intended to be used as a `value_type` in templated code.

Note: Destructor not allowed to throw.

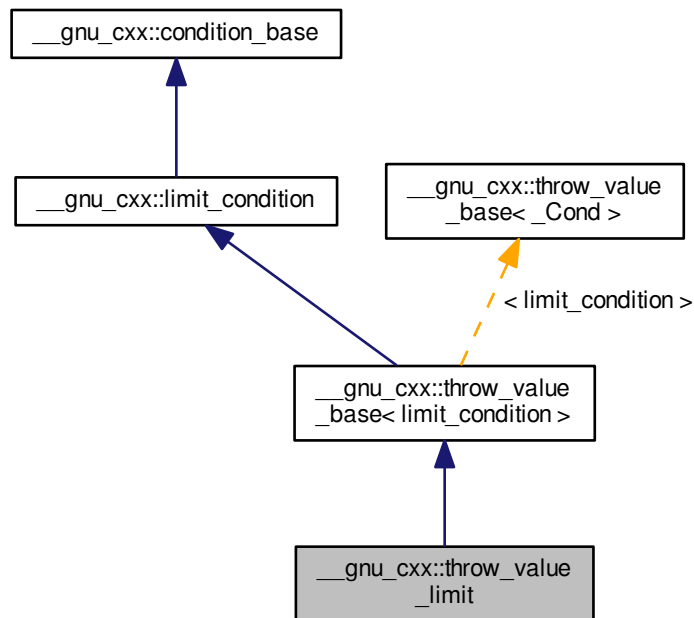
Definition at line 447 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.74 __gnu_cxx::throw_value_limit Struct Reference

Inheritance diagram for __gnu_cxx::throw_value_limit:



Public Types

- typedef [throw_value_base](#)
 < [limit_condition](#) > **base_type**
- typedef [limit_condition](#) **condition_type**

Public Member Functions

- **throw_value_limit** (const [throw_value_limit](#) &__other)
- **throw_value_limit** ([throw_value_limit](#) &&)=default
- **throw_value_limit** (const std::size_t __i)
- [throw_value_base](#) & **operator++** ()
- [throw_value_limit](#) & **operator=** (const [throw_value_limit](#) &__other)
- [throw_value_limit](#) & **operator=** ([throw_value_limit](#) &&)=default

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)

- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**

4.74.1 Detailed Description

Type throwing via limit condition.

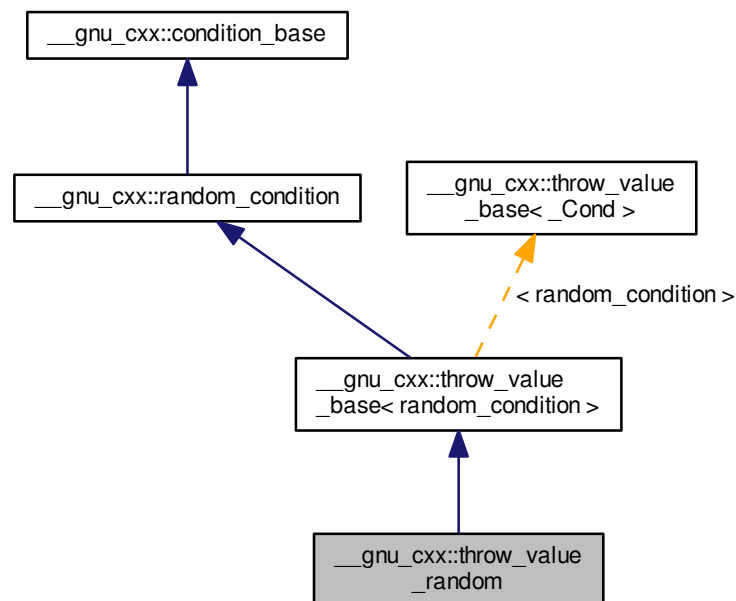
Definition at line 564 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.75 __gnu_cxx::throw_value_random Struct Reference

Inheritance diagram for __gnu_cxx::throw_value_random:



Public Types

- typedef [throw_value_base](#)
< [random_condition](#) > **base_type**
- typedef [random_condition](#) **condition_type**

Public Member Functions

- `throw_value_random` (const [throw_value_random](#) &__other)
- `throw_value_random` ([throw_value_random](#) &&)=default
- `throw_value_random` (const std::size_t __i)
- `throw_value_base` & `operator++` ()
- `throw_value_random` & `operator=` (const [throw_value_random](#) &__other)
- `throw_value_random` & `operator=` ([throw_value_random](#) &&)=default
- void `seed` (unsigned long __s)

Static Public Member Functions

- static void `set_probability` (double __p)
- static void `throw_conditionally` ()

Public Attributes

- std::size_t `_M_i`

4.75.1 Detailed Description

Type throwing via random condition.

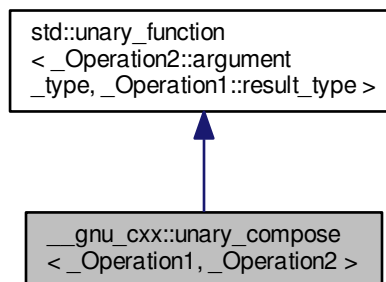
Definition at line 595 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.76 `__gnu_cxx::unary_compose< _Operation1, _Operation2 >` Class Template Reference

Inheritance diagram for `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **unary_compose** (const `_Operation1` &__x, const `_Operation2` &__y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &__x) const

Protected Attributes

- `_Operation1` **_M_fn1**
- `_Operation2` **_M_fn2**

4.76.1 Detailed Description

template<class `_Operation1`, class `_Operation2`>class `__gnu_cxx::unary_compose<_Operation1, _Operation2>`

An [SGI extension](#) .

Definition at line 125 of file `ext/functional`.

4.76.2 Member Typedef Documentation

4.76.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function<_Arg, _Result>::argument_type`
[[inherited](#)]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.76.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function<_Arg, _Result>::result_type`
[[inherited](#)]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

4.77 `__gnu_debug::After_nth_from<_Iterator>` Class Template Reference

Public Member Functions

- **After_nth_from** (const `difference_type` &__n, const `_Iterator` &[__base](#))
- bool **operator()** (const `_Iterator` &__x) const

4.77.1 Detailed Description

```
template<typename _Iterator>class __gnu_debug::_After_nth_from< _Iterator >
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 77 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

4.78 `__gnu_debug::_BeforeBeginHelper< _Sequence >` Struct Template Reference

Public Types

- typedef `_It::iterator_type` **_Baselt**
- typedef `_Sequence::const_iterator` **_It**

Static Public Member Functions

- static bool **_S_Is** (`_Baselt`, const `_Sequence *`)
- static bool **_S_Is_Beginnest** (`_Baselt` `__it`, const `_Sequence *` `__seq`)

4.78.1 Detailed Description

```
template<typename _Sequence>struct __gnu_debug::_BeforeBeginHelper< _Sequence >
```

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 45 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe_iterator.h](#)

4.79 `__gnu_debug::_Equal_to< _Type >` Class Template Reference

Public Member Functions

- **_Equal_to** (const `_Type` & `__v`)
- bool **operator()** (const `_Type` & `__x`) const

4.79.1 Detailed Description

```
template<typename _Type>class __gnu_debug::_Equal_to< _Type >
```

A simple function object that returns true if the passed-in value is equal to the stored value.

Definition at line 62 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

4.80 `__gnu_debug::Not_equal_to<_Type>` Class Template Reference

Public Member Functions

- `_Not_equal_to` (const `_Type` &__v)
- `bool operator()` (const `_Type` &__x) const

4.80.1 Detailed Description

```
template<typename _Type>class __gnu_debug::Not_equal_to<_Type>
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

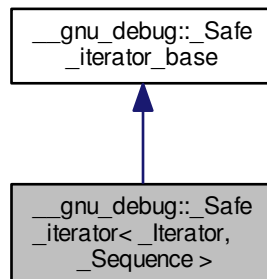
Definition at line 47 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

4.81 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_iterator<_Iterator, _Sequence>`:



Public Types

- `typedef _Traits::difference_type` **difference_type**
- `typedef _Traits::iterator_category` **iterator_category**
- `typedef _Iterator` **iterator_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::value_type` **value_type**

Public Member Functions

- `_Safe_iterator` ()
- `_Safe_iterator` (const `_Iterator` &__i, const `_Sequence` *__seq)
- `_Safe_iterator` (const `_Safe_iterator` &__x)
- `_Safe_iterator` (`_Safe_iterator` &&__x)
- `template<typename _MutableIterator >`
`_Safe_iterator` (const `_Safe_iterator`< `_MutableIterator`, typename `__gnu_cxx::__enable_if`<(std::__are_same<
`_MutableIterator`, typename `_Sequence::iterator::iterator_type` >::__value), `_Sequence` >::__type > &__x)
- `void` `_M_attach` (`_Safe_sequence_base` *__seq, bool __constant)
- `void` `_M_attach` (`_Safe_sequence_base` *__seq)
- `void` `_M_attach_single` (`_Safe_sequence_base` *__seq, bool __constant) throw ()
- `void` `_M_attach_single` (`_Safe_sequence_base` *__seq)
- `bool` `_M_attached_to` (const `_Safe_sequence_base` *__seq) const
- `bool` `_M_before_dereferenceable` () const
- `bool` `_M_can_advance` (const `difference_type` &__n) const
- `bool` `_M_can_compare` (const `_Safe_iterator_base` &__x) const throw ()
- `bool` `_M_decrementable` () const
- `bool` `_M_dereferenceable` () const
- `void` `_M_detach` ()
- `void` `_M_detach_single` () throw ()
- `const _Sequence *` `_M_get_sequence` () const
- `bool` `_M_incrementable` () const
- `void` `_M_invalidate` ()
- `bool` `_M_is_before_begin` () const
- `bool` `_M_is_begin` () const
- `bool` `_M_is_beginnest` () const
- `bool` `_M_is_end` () const
- `void` `_M_reset` () throw ()
- `bool` `_M_singular` () const throw ()
- `void` `_M_unlink` () throw ()
- `template<typename _Other >`
`bool` `_M_valid_range` (const `_Safe_iterator`< `_Other`, `_Sequence` > &__rhs) const
- `_Iterator base` () const
- `operator _Iterator` () const
- `reference operator*` () const
- `_Safe_iterator operator+` (const `difference_type` &__n) const
- `_Safe_iterator & operator++` ()
- `_Safe_iterator operator++` (int)
- `_Safe_iterator & operator+=` (const `difference_type` &__n)
- `_Safe_iterator operator-` (const `difference_type` &__n) const
- `_Safe_iterator & operator--` ()
- `_Safe_iterator operator--` (int)
- `_Safe_iterator & operator-=` (const `difference_type` &__n)
- `pointer operator->` () const
- `_Safe_iterator & operator=` (const `_Safe_iterator` &__x)
- `_Safe_iterator & operator=` (`_Safe_iterator` &&__x)
- `reference operator[]` (const `difference_type` &__n) const

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

Protected Member Functions

- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

4.81.1 Detailed Description

```
template<typename _Iterator, typename _Sequence> class __gnu_debug::_Safe_iterator< _Iterator, _Sequence >
```

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 116 of file `safe_iterator.h`.

4.81.2 Constructor & Destructor Documentation

```
4.81.2.1 template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::_Safe_iterator ( ) [inline]
```

Postcondition

the iterator is singular and unattached

Definition at line 142 of file `safe_iterator.h`.

```
4.81.2.2 template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::_Safe_iterator ( const _Iterator & __i, const _Sequence * __seq ) [inline]
```

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 151 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

4.81.2.3 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator (const _Safe_iterator<_Iterator, _Sequence> & __x) [inline]`

Copy construction.

Definition at line 162 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

4.81.2.4 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator (_Safe_iterator<_Iterator, _Sequence> && __x) [inline]`

Move construction.

Postcondition

`__x` is singular and unattached

Definition at line 179 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach()`.

4.81.2.5 `template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator (const _Safe_iterator<_MutableIterator, typename __gnu_cxx::__enable_if<(std::__are_same<_MutableIterator, typename _Sequence::iterator::iterator_type>::value), _Sequence>::type > & __x) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 197 of file `safe_iterator.h`.

4.81.3 Member Function Documentation

4.81.3.1 `void __gnu_debug::_Safe_iterator_base::_M_attach (_Safe_sequence_base * __seq, bool __constant) [inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_attach()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach()`, and `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`.

4.81.3.2 `template<typename _Iterator, typename _Sequence> void __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach (_Safe_sequence_base * __seq) [inline]`

Attach iterator to the given sequence.

Definition at line 406 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_attach()`.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator=()`.

4.81.3.3 `void __gnu_debug::_Safe_iterator_base::_M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw () [inherited]`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_attach_single()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_attach_single()`.

4.81.3.4 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_attach_single (_Safe_sequence_base * __seq) [inline]`

Likewise, but not thread-safe.

Definition at line 413 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator_base::_M_attach_single()`.

4.81.3.5 `bool __gnu_debug::Safe_iterator_base::_M_attached_to (const _Safe_sequence_base * __seq) const [inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::_M_sequence`.

4.81.3.6 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_before_dereferenceable () const [inline]`

Is the iterator before a dereferenceable one?

Definition at line 425 of file `safe_iterator.h`.

References `__gnu_debug::_base()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

4.81.3.7 `bool __gnu_debug::Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw () [inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.81.3.8 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable () const [inline]`

Is the iterator dereferenceable?

Definition at line 420 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_is_before_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::check_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator*()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator->()`.

4.81.3.9 `void __gnu_debug::Safe_iterator_base::_M_detach () [inherited]`

Detach the iterator for whatever sequence it is attached to, if any.

4.81.3.10 `void __gnu_debug::Safe_iterator_base::_M_detach_single () throw () [inherited]`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

4.81.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex () throw ()` [protected],
[inherited]

For use in `_Safe_iterator`.

4.81.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable () const` [inline]

Is the iterator incrementable?

Definition at line 437 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end()`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_before_dereferenceable()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++()`.

4.81.3.13 `void __gnu_debug::Safe_iterator_base::M_invalidate ()` [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_version`.

4.81.3.14 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_before_begin () const` [inline]

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Definition at line 468 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

4.81.3.15 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_begin () const` [inline]

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 459 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

4.81.3.16 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_beginnest () const` [inline]

Is this iterator equal to the sequence's `before_begin()` iterator if any or `begin()` otherwise?

Definition at line 475 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

4.81.3.17 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end () const` [inline]

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 463 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.81.3.18 `void __gnu_debug::Safe_iterator_base::M_reset () throw ()` `[inherited]`

Reset all member variables

4.81.3.19 `bool __gnu_debug::Safe_iterator_base::M_singular () const throw ()` `[inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::__check_singular()`, `__gnu_debug::__check_singular_aux()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_incrementable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::Safe_iterator()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::Safe_local_iterator()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator=()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=()`.

4.81.3.20 `void __gnu_debug::Safe_iterator_base::M_unlink () throw ()` `[inline], [inherited]`

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_next`, and `__gnu_debug::Safe_iterator_base::M_prior`.

4.81.3.21 `template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::Safe_iterator<_Iterator, _Sequence>::base () const` `[inline]`

Return the underlying iterator.

Definition at line 396 of file `safe_iterator.h`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_before_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_before_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_beginnest()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_end()`.

4.81.3.22 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator _Iterator () const` `[inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 402 of file `safe_iterator.h`.

4.81.3.23 `template<typename _Iterator, typename _Sequence> reference __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator*() const` `[inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 260 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

4.81.3.24 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ () [inline]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 289 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

4.81.3.25 `template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ (int) [inline]`

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 303 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

4.81.3.26 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator-- () [inline]`

Iterator predecrement.

Precondition

iterator is decrementable

Definition at line 319 of file `safe_iterator.h`.

4.81.3.27 `template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator-- (int) [inline]`

Iterator postdecrement.

Precondition

iterator is decrementable

Definition at line 333 of file `safe_iterator.h`.

4.81.3.28 `template<typename _Iterator, typename _Sequence> pointer __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator-> () const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Todo Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Definition at line 275 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

4.81.3.29 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=(const _Safe_iterator<_Iterator, _Sequence> &_x) [inline]`

Copy assignment.

Definition at line 217 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_attach()`, `__gnu_debug::Safe_iterator_base::M_sequence`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

4.81.3.30 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=(_Safe_iterator<_Iterator, _Sequence> &&_x) [inline]`

Move assignment.

Postcondition

`__x` is singular and unattached

Definition at line 237 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_attach()`.

4.81.4 Member Data Documentation

4.81.4.1 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_next [inherited]`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`, and `__gnu_debug::Safe_iterator_base::M_unlink()`.

4.81.4.2 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_prior [inherited]`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`, and `__gnu_debug::Safe_iterator_base::M_unlink()`.

4.81.4.3 `_Safe_sequence_base* __gnu_debug::Safe_iterator_base::M_sequence [inherited]`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by __gnu_debug::_Safe_iterator_base::_M_attached_to(), __gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if(), __gnu_debug::_Safe_iterator_base::_Safe_iterator_base(), __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base(), __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator=(), and __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator=().

4.81.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Definition at line 64 of file safe_base.h.

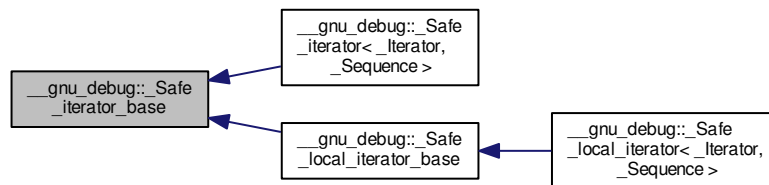
Referenced by __gnu_debug::_Safe_iterator_base::_M_invalidate().

The documentation for this class was generated from the following files:

- [safe_iterator.h](#)
- [safe_iterator.tcc](#)

4.82 __gnu_debug::_Safe_iterator_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- void [_M_attach](#) ([_Safe_sequence_base](#) * __seq, bool __constant)
- void [_M_attach_single](#) ([_Safe_sequence_base](#) * __seq, bool __constant) throw ()
- bool [_M_attached_to](#) (const [_Safe_sequence_base](#) * __seq) const
- bool [_M_can_compare](#) (const [_Safe_iterator_base](#) & __x) const throw ()
- void [_M_detach](#) ()
- void [_M_detach_single](#) () throw ()
- void [_M_invalidate](#) ()
- void [_M_reset](#) () throw ()
- bool [_M_singular](#) () const throw ()
- void [_M_unlink](#) () throw ()

Public Attributes

- [_Safe_iterator_base](#) * [_M_next](#)

- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

Protected Member Functions

- `_Safe_iterator_base ()`
- `_Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant)`
- `_Safe_iterator_base (const _Safe_iterator_base &)`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `_Safe_iterator_base & operator= (const _Safe_iterator_base &)`

4.82.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_base.h`.

4.82.2 Constructor & Destructor Documentation

4.82.2.1 `__gnu_debug::Safe_iterator_base::Safe_iterator_base () [inline], [protected]`

Initializes the iterator and makes it singular.

Definition at line 76 of file `safe_base.h`.

4.82.2.2 `__gnu_debug::Safe_iterator_base::Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant) [inline], [protected]`

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 87 of file `safe_base.h`.

References `_M_attach()`.

4.82.2.3 `__gnu_debug::Safe_iterator_base::Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant) [inline], [protected]`

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 94 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.

4.82.3 Member Function Documentation

4.82.3.1 void __gnu_debug::Safe_iterator_base::M_attach (_Safe_sequence_base * __seq, bool __constant)

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach(), and _Safe_iterator_base().

4.82.3.2 void __gnu_debug::Safe_iterator_base::M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()

Likewise, but not thread-safe.

Referenced by __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach_single(), and __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single().

4.82.3.3 bool __gnu_debug::Safe_iterator_base::M_attached_to (const _Safe_sequence_base * __seq) const [inline]

Determines if we are attached to the given sequence.

Definition at line 129 of file safe_base.h.

References _M_sequence.

4.82.3.4 bool __gnu_debug::Safe_iterator_base::M_can_compare (const _Safe_iterator_base & __x) const throw ()

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.

4.82.3.5 void __gnu_debug::Safe_iterator_base::M_detach ()

Detach the iterator for whatever sequence it is attached to, if any.

4.82.3.6 void __gnu_debug::Safe_iterator_base::M_detach_single () throw ()

Likewise, but not thread-safe.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if().

4.82.3.7 __gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex () throw () [protected]

For use in _Safe_iterator.

4.82.3.8 void __gnu_debug::Safe_iterator_base::M_invalidate () [inline]

Invalidate the iterator, making it singular.

Definition at line 142 of file safe_base.h.

References _M_version.

4.82.3.9 void __gnu_debug::Safe_iterator_base::M_reset () throw ()

Reset all member variables

4.82.3.10 bool __gnu_debug::Safe_iterator_base::M_singular () const throw ()

Is this iterator singular?

Referenced by __gnu_debug::__check_singular(), __gnu_debug::__check_singular_aux(), __gnu_debug::Safe_local-

```

_iterator< _Iterator, _Sequence >::M_dereferenceable(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_dereferenceable(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_incrementable(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_incrementable(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::Safe_iterator(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::Safe_local_iterator(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator=().

```

4.82.3.11 `void __gnu_debug::Safe_iterator_base::M_unlink () throw () [inline]`

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `_M_next`, and `_M_prior`.

4.82.4 Member Data Documentation

4.82.4.1 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`, and `_M_unlink()`.

4.82.4.2 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`, and `_M_unlink()`.

4.82.4.3 `_Safe_sequence_base* __gnu_debug::Safe_iterator_base::M_sequence`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `_M_attached_to()`, `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`, `_Safe_iterator_base()`, `__gnu_debug::Safe_local_iterator_base::Safe_local_iterator_base()`, `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::operator=()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator=()`.

4.82.4.4 `unsigned int __gnu_debug::Safe_iterator_base::M_version`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 64 of file `safe_base.h`.

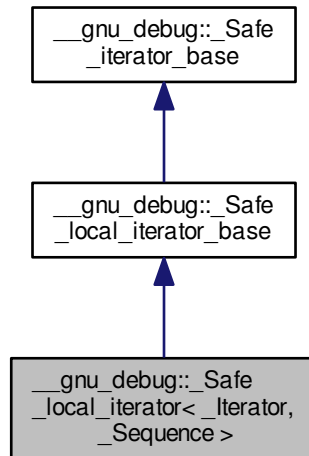
Referenced by `_M_invalidate()`.

The documentation for this class was generated from the following file:

- [safe_base.h](#)

4.83 `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>`:



Public Types

- typedef `_Traits::difference_type` **difference_type**
- typedef `_Traits::iterator_category` **iterator_category**
- typedef `_Iterator` **iterator_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value_type**

Public Member Functions

- `_Safe_local_iterator ()`
- `_Safe_local_iterator (const _Iterator &__i, size_type __bucket, const _Sequence *__seq)`
- `_Safe_local_iterator (const _Safe_local_iterator &__x)`
- `template<typename _MutableIterator > _Safe_local_iterator (const _Safe_local_iterator<_MutableIterator, typename __gnu_cxx::__enable_if<std::__are_same<_MutableIterator, typename _Sequence::local_iterator::iterator_type>::__value, _Sequence>::__type> &__x)`
- `void _M_attach (_Safe_sequence_base *__seq, bool __constant)`
- `void _M_attach (_Safe_sequence_base *__seq)`
- `void _M_attach_single (_Safe_sequence_base *__seq, bool __constant) throw ()`
- `void _M_attach_single (_Safe_sequence_base *__seq)`
- `bool _M_attached_to (const _Safe_sequence_base *__seq) const`
- `bool _M_can_compare (const _Safe_iterator_base &__x) const throw ()`
- `bool _M_dereferenceable () const`

- `void _M_detach ()`
- `void _M_detach_single () throw ()`
- `const _Sequence * _M_get_sequence () const`
- `template<typename _Other>`
`bool _M_in_same_bucket (const _Safe_local_iterator<_Other, _Sequence> &__other) const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `void _M_reset () throw ()`
- `bool _M_singular () const throw ()`
- `void _M_unlink () throw ()`
- `template<typename _Other>`
`bool _M_valid_range (const _Safe_local_iterator<_Other, _Sequence> &__rhs) const`
- `_Iterator base () const`
- `size_type bucket () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_local_iterator & operator++ ()`
- `_Safe_local_iterator operator++ (int)`
- `pointer operator-> () const`
- `_Safe_local_iterator & operator= (const _Safe_local_iterator &__x)`

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

Protected Member Functions

- `_Safe_unordered_container_base * _M_get_container () const noexcept`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

4.83.1 Detailed Description

`template<typename _Iterator, typename _Sequence> class __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>`

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 52 of file `safe_local_iterator.h`.

4.83.2 Constructor & Destructor Documentation

4.83.2.1 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator () [inline]`

Postcondition

the iterator is singular and unattached

Definition at line 82 of file `safe_local_iterator.h`.

4.83.2.2 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator (const _Iterator & __i, size_type __bucket, const _Sequence * __seq) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 91 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_singular()`.

4.83.2.3 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator (const _Safe_local_iterator<_Iterator, _Sequence> & __x) [inline]`

Copy construction.

Definition at line 104 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_singular()`.

4.83.2.4 `template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator (const _Safe_local_iterator<_MutableIterator, typename __gnu_cxx::__enable_if<std::__are_same<_MutableIterator, typename _Sequence::local_iterator::iterator_type>::__value, _Sequence>::__type> & __x) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 122 of file `safe_local_iterator.h`.

4.83.3 Member Function Documentation

4.83.3.1 `void __gnu_debug::Safe_local_iterator_base::M_attach (_Safe_sequence_base * __seq, bool __constant) [inherited]`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_local_iterator_base::Safe_local_iterator_base()`.

4.83.3.2 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_attach (_Safe_sequence_base * __seq) [inline]`

Attach iterator to the given sequence.

Definition at line 238 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::_M_attach()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator=()`.

4.83.3.3 `void __gnu_debug::Safe_iterator_base::_M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw () [inherited]`

Likewise, but not thread-safe.

4.83.3.4 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_attach_single (_Safe_sequence_base * __seq) [inline]`

Likewise, but not thread-safe.

Definition at line 243 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::_M_attach_single()`.

4.83.3.5 `bool __gnu_debug::Safe_iterator_base::_M_attached_to (const _Safe_sequence_base * __seq) const [inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::_M_sequence`.

4.83.3.6 `bool __gnu_debug::Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw () [inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.83.3.7 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable () const [inline]`

Is the iterator dereferenceable?

Definition at line 248 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator*()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator->()`.

4.83.3.8 `void __gnu_debug::Safe_local_iterator_base::_M_detach () [inherited]`

Detach the iterator for whatever container it is attached to, if any.

4.83.3.9 `void __gnu_debug::Safe_local_iterator_base::_M_detach_single () throw () [inherited]`

Likewise, but not thread-safe.

4.83.3.10 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::M_get_mutex () throw ()` [protected],
[inherited]

For use in `_Safe_iterator`.

4.83.3.11 `template<typename _Iterator, typename _Sequence> template<typename _Other> bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_in_same_bucket (const _Safe_local_iterator<_Other, _Sequence> & _other) const` [inline]

Is this iterator part of the same bucket as the other one?

Definition at line 277 of file `safe_local_iterator.h`.

4.83.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_incrementable () const` [inline]

Is the iterator incrementable?

Definition at line 253 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_is_end()`, and `__gnu_debug::_Safe_iterator_base::M_singular()`.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator++()`.

4.83.3.13 `void __gnu_debug::_Safe_iterator_base::M_invalidate ()` [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::M_version`.

4.83.3.14 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_is_begin () const` [inline]

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 268 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::base()`.

4.83.3.15 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_is_end () const` [inline]

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 272 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_dereferenceable()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.83.3.16 `void __gnu_debug::_Safe_iterator_base::M_reset () throw ()` [inherited]

Reset all member variables

4.83.3.17 `bool __gnu_debug::_Safe_iterator_base::M_singular () const throw ()` [inherited]

Is this iterator singular?

Referenced by `__gnu_debug::__check_singular()`, `__gnu_debug::__check_singular_aux()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::__M_dereferenceable()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_incrementable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::__M_incrementable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::__Safe_iterator()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator=()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=()`.

4.83.3.18 `void __gnu_debug::Safe_iterator_base::M_unlink () throw () [inline], [inherited]`

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_next`, and `__gnu_debug::Safe_iterator_base::M_prior`.

4.83.3.19 `template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::base () const [inline]`

Return the underlying iterator.

Definition at line 222 of file `safe_local_iterator.h`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_is_begin()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_is_end()`.

4.83.3.20 `template<typename _Iterator, typename _Sequence> size_type __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::bucket () const [inline]`

Return the bucket.

Definition at line 228 of file `safe_local_iterator.h`.

4.83.3.21 `template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator _Iterator () const [inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 234 of file `safe_local_iterator.h`.

4.83.3.22 `template<typename _Iterator, typename _Sequence> reference __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator* () const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 164 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::__M_dereferenceable()`.

4.83.3.23 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator++ () [inline]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 193 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.83.3.24 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator++(int) [inline]`

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 207 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.83.3.25 `template<typename _Iterator, typename _Sequence> pointer __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator->() const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Todo Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Definition at line 179 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

4.83.3.26 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator=(const _Safe_local_iterator<_Iterator, _Sequence> &_x) [inline]`

Copy assignment.

Definition at line 144 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_attach()`, `__gnu_debug::Safe_iterator_base::M_sequence`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

4.83.4 Member Data Documentation

4.83.4.1 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_next [inherited]`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`, and `__gnu_debug::Safe_iterator_base::M_unlink()`.

4.83.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

4.83.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator=()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator=()`.

4.83.4.4 `unsigned int __gnu_debug::_Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 64 of file `safe_base.h`.

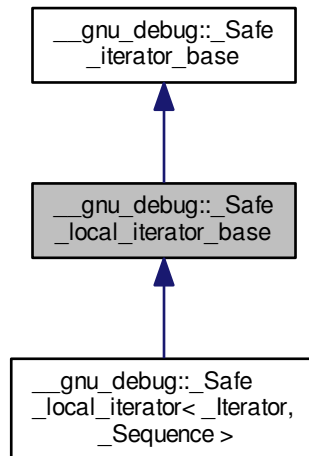
Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`.

The documentation for this class was generated from the following files:

- [safe_local_iterator.h](#)
- [safe_local_iterator.tcc](#)

4.84 __gnu_debug::_Safe_local_iterator_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_local_iterator_base:



Public Member Functions

- void [_M_attach](#) ([_Safe_sequence_base](#) * __seq, bool __constant)
- void [_M_attach_single](#) ([_Safe_sequence_base](#) * __seq, bool __constant) throw ()
- bool [_M_attached_to](#) (const [_Safe_sequence_base](#) * __seq) const
- bool [_M_can_compare](#) (const [_Safe_iterator_base](#) & __x) const throw ()
- void [_M_detach](#) ()
- void [_M_detach_single](#) () throw ()
- void [_M_invalidate](#) ()
- void [_M_reset](#) () throw ()
- bool [_M_singular](#) () const throw ()
- void [_M_unlink](#) () throw ()

Public Attributes

- [_Safe_iterator_base](#) * [_M_next](#)
- [_Safe_iterator_base](#) * [_M_prior](#)
- [_Safe_sequence_base](#) * [_M_sequence](#)
- unsigned int [_M_version](#)

Protected Member Functions

- [_Safe_local_iterator_base](#) ()
- [_Safe_local_iterator_base](#) (const [_Safe_sequence_base](#) * __seq, bool __constant)

- `_Safe_local_iterator_base` (const `_Safe_local_iterator_base` & __x, bool __constant)
- `_Safe_local_iterator_base` (const `_Safe_local_iterator_base` &)
- `_Safe_unordered_container_base * _M_get_container` () const noexcept
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `_Safe_local_iterator_base & operator=` (const `_Safe_local_iterator_base` &)

4.84.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_unordered_base.h`.

4.84.2 Constructor & Destructor Documentation

4.84.2.1 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ()` `[inline]`, `[protected]`

Initializes the iterator and makes it singular.

Definition at line 54 of file `safe_unordered_base.h`.

4.84.2.2 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (const _Safe_sequence_base * __seq, bool __constant)` `[inline]`, `[protected]`

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 64 of file `safe_unordered_base.h`.

References `_M_attach()`.

4.84.2.3 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (const _Safe_local_iterator_base & __x, bool __constant)` `[inline]`, `[protected]`

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 70 of file `safe_unordered_base.h`.

References `_M_attach()`, and `__gnu_debug::_Safe_iterator_base::_M_sequence`.

4.84.3 Member Function Documentation

4.84.3.1 `void __gnu_debug::_Safe_local_iterator_base::_M_attach (_Safe_sequence_base * __seq, bool __constant)`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `_Safe_local_iterator_base()`.

4.84.3.2 void __gnu_debug::Safe_local_iterator_base::M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()

Likewise, but not thread-safe.

4.84.3.3 bool __gnu_debug::Safe_iterator_base::M_attached_to (const _Safe_sequence_base * __seq) const [inline], [inherited]

Determines if we are attached to the given sequence.

Definition at line 129 of file safe_base.h.

References __gnu_debug::Safe_iterator_base::M_sequence.

4.84.3.4 bool __gnu_debug::Safe_iterator_base::M_can_compare (const _Safe_iterator_base & __x) const throw () [inherited]

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.

4.84.3.5 void __gnu_debug::Safe_local_iterator_base::M_detach ()

Detach the iterator for whatever container it is attached to, if any.

4.84.3.6 void __gnu_debug::Safe_local_iterator_base::M_detach_single () throw ()

Likewise, but not thread-safe.

4.84.3.7 __gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex () throw () [protected], [inherited]

For use in _Safe_iterator.

4.84.3.8 void __gnu_debug::Safe_iterator_base::M_invalidate () [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file safe_base.h.

References __gnu_debug::Safe_iterator_base::M_version.

4.84.3.9 void __gnu_debug::Safe_iterator_base::M_reset () throw () [inherited]

Reset all member variables

4.84.3.10 bool __gnu_debug::Safe_iterator_base::M_singular () const throw () [inherited]

Is this iterator singular?

Referenced by __gnu_debug::__check_singular(), __gnu_debug::__check_singular_aux(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_dereferenceable(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_dereferenceable(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_incrementable(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_incrementable(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::Safe_iterator(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::Safe_local_iterator(), __gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator=().

4.84.3.11 void __gnu_debug::Safe_iterator_base::M_unlink () throw () [inline], [inherited]

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_next`, and `__gnu_debug::Safe_iterator_base::M_prior`.

4.84.4 Member Data Documentation

4.84.4.1 `Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_next` [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`, and `__gnu_debug::Safe_iterator_base::M_unlink()`.

4.84.4.2 `Safe_iterator_base* __gnu_debug::Safe_iterator_base::M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`, and `__gnu_debug::Safe_iterator_base::M_unlink()`.

4.84.4.3 `Safe_sequence_base* __gnu_debug::Safe_iterator_base::M_sequence` [inherited]

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator_base::M_attached_to()`, `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`, `__gnu_debug::Safe_iterator_base::Safe_iterator_base()`, `Safe_local_iterator_base()`, `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::operator=()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator=()`.

4.84.4.4 `unsigned int __gnu_debug::Safe_iterator_base::M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 64 of file `safe_base.h`.

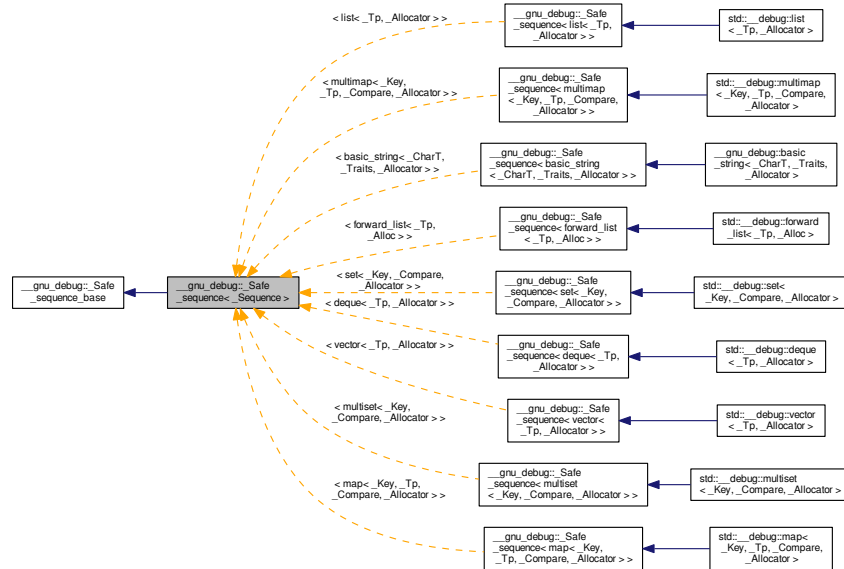
Referenced by `__gnu_debug::Safe_iterator_base::M_invalidate()`.

The documentation for this class was generated from the following file:

- [safe_unordered_base.h](#)

4.85 `__gnu_debug::Safe_sequence<_Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_sequence<_Sequence>`:



Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`
- `template<typename _Predicate> void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate> void _M_transfer_from_if (_Safe_sequence & __from, _Predicate __pred)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base & __x)`

4.85.1 Detailed Description

```
template<typename _Sequence>class __gnu_debug::_Safe_sequence<_Sequence>
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 111 of file `safe_sequence.h`.

4.85.2 Member Function Documentation

4.85.2.1 `void __gnu_debug::_Safe_sequence_base::M.attach (_Safe_iterator_base * __it, bool __constant)` [inherited]

Attach an iterator to this sequence.

4.85.2.2 `void __gnu_debug::_Safe_sequence_base::M.attach_single (_Safe_iterator_base * __it, bool __constant) throw ()` [inherited]

Likewise but not thread safe.

4.85.2.3 `void __gnu_debug::_Safe_sequence_base::M.detach (_Safe_iterator_base * __it)` [inherited]

Detach an iterator from this sequence

4.85.2.4 `void __gnu_debug::_Safe_sequence_base::M.detach_all ()` [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base()`.

4.85.2.5 `void __gnu_debug::_Safe_sequence_base::M.detach_single (_Safe_iterator_base * __it) throw ()` [inherited]

Likewise but not thread safe.

4.85.2.6 `void __gnu_debug::_Safe_sequence_base::M.detach_singular ()` [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

4.85.2.7 `__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::M.get_mutex () throw ()` [protected], [inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.85.2.8 `void __gnu_debug::_Safe_sequence_base::M.invalidate_all () const` [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

References `__gnu_debug::Safe_sequence_base::M_version`.

4.85.2.9 `template<typename _Sequence> template<typename _Predicate> void _Safe_sequence<_Sequence>::M_invalidate_if (_Predicate __pred)`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

4.85.2.10 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.85.2.11 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x)` `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.85.2.12 `template<typename _Sequence> template<typename _Predicate> void _Safe_sequence<_Sequence>::M_transfer_from_if (_Safe_sequence<_Sequence> & __from, _Predicate __pred)`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

References `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_iterator_base::M_detach_single()`, `__gnu_debug::Safe_sequence_base::M_get_mutex()`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_next`, `__gnu_debug::Safe_iterator_base::M_prior`, `__gnu_debug::Safe_iterator_base::M_sequence`, and `__gnu_debug::Safe_sequence_base::M_version`.

4.85.3 Member Data Documentation

4.85.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.85.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.85.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

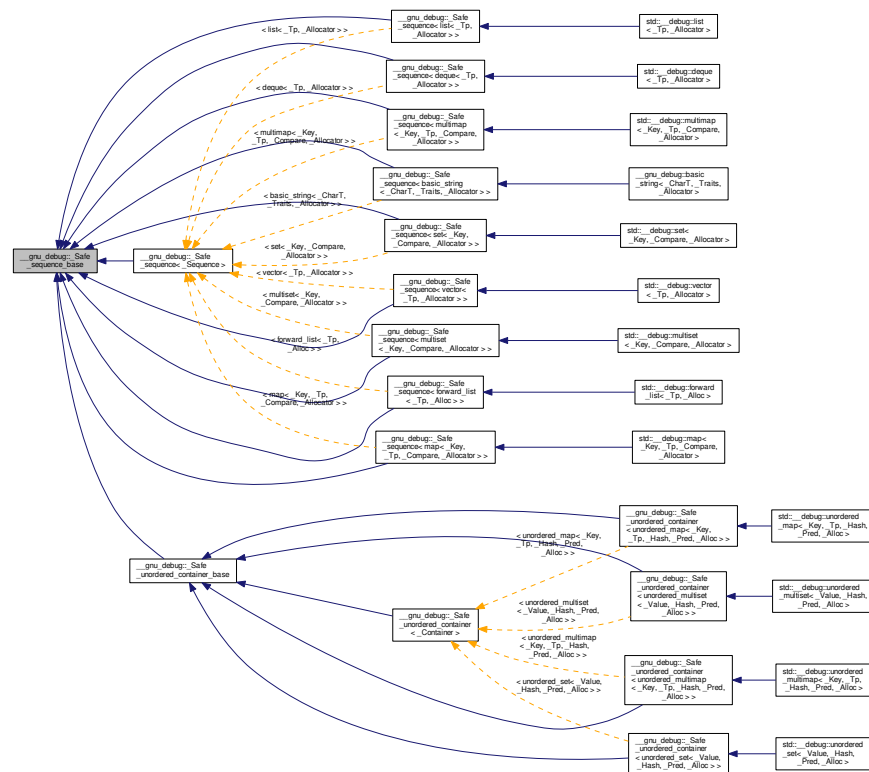
Referenced by __gnu_debug::Safe_sequence_base::M_invalidate_all(), and __gnu_debug::Safe_sequence< __-Sequence >::M_transfer_from_if().

The documentation for this class was generated from the following files:

- [safe_sequence.h](#)
- [safe_sequence.tcc](#)

4.86 __gnu_debug::Safe_sequence_base Class Reference

Inheritance diagram for __gnu_debug::Safe_sequence_base:



Public Member Functions

- void [M_attach](#) ([Safe_iterator_base](#) * __it, bool __constant)
- void [M_attach_single](#) ([Safe_iterator_base](#) * __it, bool __constant) throw ()
- void [M_detach](#) ([Safe_iterator_base](#) * __it)
- void [M_detach_single](#) ([Safe_iterator_base](#) * __it) throw ()
- void [M_invalidate_all](#) () const

Public Attributes

- [Safe_iterator_base](#) * [M_const_iterators](#)
- [Safe_iterator_base](#) * [M_iterators](#)
- unsigned int [M_version](#)

Protected Member Functions

- `~_Safe_sequence_base()`
- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

4.86.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 177 of file `safe_base.h`.

4.86.2 Constructor & Destructor Documentation

4.86.2.1 `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()` `[inline]`, `[protected]`

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 197 of file `safe_base.h`.

References `_M_detach_all()`.

4.86.3 Member Function Documentation

4.86.3.1 `void __gnu_debug::Safe_sequence_base::M.attach(_Safe_iterator_base * __it, bool __constant)`

Attach an iterator to this sequence.

4.86.3.2 `void __gnu_debug::Safe_sequence_base::M.attach_single(_Safe_iterator_base * __it, bool __constant) throw()`

Likewise but not thread safe.

4.86.3.3 `void __gnu_debug::Safe_sequence_base::M.detach(_Safe_iterator_base * __it)`

Detach an iterator from this sequence

4.86.3.4 `void __gnu_debug::Safe_sequence_base::M.detach_all()` `[protected]`

Detach all iterators, leaving them singular.

Referenced by `~_Safe_sequence_base()`.

4.86.3.5 `void __gnu_debug::Safe_sequence_base::M.detach_single(_Safe_iterator_base * __it) throw()`

Likewise but not thread safe.

4.86.3.6 `void __gnu_debug::Safe_sequence_base::M_detach_singular ()` [protected]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

4.86.3.7 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw ()` [protected]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.86.3.8 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const` [inline]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

References `_M_version`.

4.86.3.9 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` [protected]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.86.3.10 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x)` [protected]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.86.4 Member Data Documentation

4.86.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.86.4.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.86.4.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

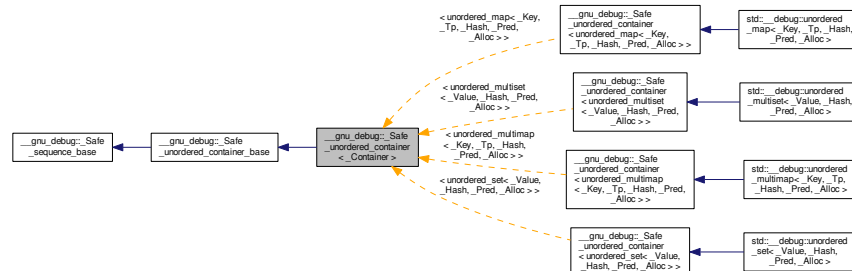
Referenced by `_M_invalidate_all()`, and `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [safe_base.h](#)

4.87 `__gnu_debug::Safe_unordered_container<_Container>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_unordered_container<_Container>`:



Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_local (_Safe_iterator_base * __it)`
- `void _M_detach_local_single (_Safe_iterator_base * __it) throw ()`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`
- `template<typename _Predicate>`
`void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate>`
`void _M_invalidate_local_if (_Predicate __pred)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_unordered_container_base & __x)`
- `void _M_swap (_Safe_sequence_base & __x)`

4.87.1 Detailed Description

```
template<typename _Container>class __gnu_debug::Safe_unordered_container<_Container>
```

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

Definition at line 58 of file `safe_unordered_container.h`.

4.87.2 Member Function Documentation

4.87.2.1 `void __gnu_debug::Safe_sequence_base::M_attach (_Safe_iterator_base * __it, bool __constant)` [inherited]

Attach an iterator to this sequence.

4.87.2.2 `void __gnu_debug::Safe_unordered_container_base::M_attach_local (_Safe_iterator_base * __it, bool __constant)`
[inherited]

Attach an iterator to this container.

4.87.2.3 `void __gnu_debug::Safe_unordered_container_base::M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw ()` [inherited]

Likewise but not thread safe.

4.87.2.4 `void __gnu_debug::Safe_sequence_base::M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
[inherited]

Likewise but not thread safe.

4.87.2.5 `void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it)` [inherited]

Detach an iterator from this sequence

4.87.2.6 `void __gnu_debug::Safe_unordered_container_base::M_detach_all ()` [protected], [inherited]

Detach all iterators, leaving them singular.

4.87.2.7 `void __gnu_debug::Safe_unordered_container_base::M_detach_local (_Safe_iterator_base * __it)`
[inherited]

Detach an iterator from this container

4.87.2.8 `void __gnu_debug::Safe_unordered_container_base::M_detach_local_single (_Safe_iterator_base * __it) throw ()`
[inherited]

Likewise but not thread safe.

4.87.2.9 `void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw ()` [inherited]

Likewise but not thread safe.

4.87.2.10 `void __gnu_debug::Safe_sequence_base::M_detach_singular ()` [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

4.87.2.11 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw ()` [protected], [inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.87.2.12 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const` [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

References `__gnu_debug::Safe_sequence_base::_M_version`.

4.87.2.13 `template<typename _Container> template<typename _Predicate> void __gnu_debug::Safe_unordered_container<_Container>::M_invalidate_if (_Predicate __pred)`

Invalidates all iterators *x* that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

4.87.2.14 `template<typename _Container> template<typename _Predicate> void __gnu_debug::Safe_unordered_container<_Container>::M_invalidate_local_if (_Predicate __pred)`

Invalidates all local iterators *x* that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

4.87.2.15 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.87.2.16 `void __gnu_debug::Safe_unordered_container_base::M_swap (_Safe_unordered_container_base & __x)` [protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.87.2.17 `void __gnu_debug::Safe_sequence_base::M.swap (_Safe_sequence_base & __x)` [protected],
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.87.3 Member Data Documentation

4.87.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M.const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.87.3.2 `_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M.const_local_iterators` [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file `safe_unordered_base.h`.

4.87.3.3 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M.iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.87.3.4 `_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M.local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

4.87.3.5 `unsigned int __gnu_debug::Safe_sequence_base::M.version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

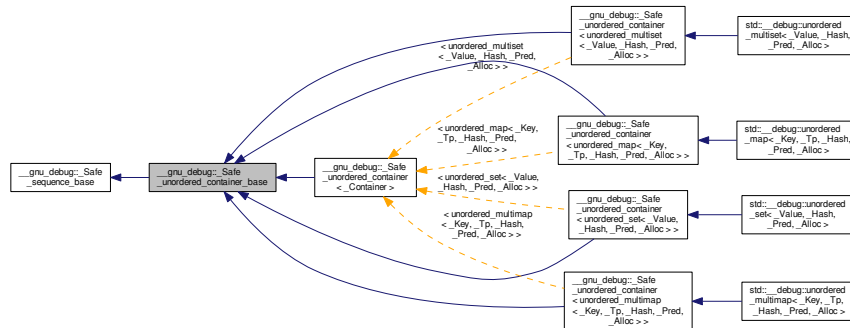
Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

The documentation for this class was generated from the following files:

- [safe_unordered_container.h](#)
- [safe_unordered_container.tcc](#)

4.88 __gnu_debug::Safe_unordered_container_base Class Reference

Inheritance diagram for __gnu_debug::Safe_unordered_container_base:



Public Member Functions

- void [_M_attach](#) ([_Safe_iterator_base](#) * __it, bool __constant)
- void [_M_attach_local](#) ([_Safe_iterator_base](#) * __it, bool __constant)
- void [_M_attach_local_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw ()
- void [_M_attach_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw ()
- void [_M_detach](#) ([_Safe_iterator_base](#) * __it)
- void [_M_detach_local](#) ([_Safe_iterator_base](#) * __it)
- void [_M_detach_local_single](#) ([_Safe_iterator_base](#) * __it) throw ()
- void [_M_detach_single](#) ([_Safe_iterator_base](#) * __it) throw ()
- void [_M_invalidate_all](#) () const

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_const_local_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- [_Safe_iterator_base](#) * [_M_local_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- [~_Safe_unordered_container_base](#) ()
- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_unordered_container_base](#) & __x)
- void [_M_swap](#) ([_Safe_sequence_base](#) & __x)

4.88.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 123 of file `safe_unordered_base.h`.

4.88.2 Constructor & Destructor Documentation

4.88.2.1 `__gnu_debug::Safe_unordered_container_base::~~Safe_unordered_container_base ()` `[inline]`, `[protected]`

Notify all iterators that reference this container that the container is being destroyed.

Definition at line 141 of file `safe_unordered_base.h`.

4.88.3 Member Function Documentation

4.88.3.1 `void __gnu_debug::Safe_sequence_base::M_attach (_Safe_iterator_base * __it, bool __constant)` `[inherited]`

Attach an iterator to this sequence.

4.88.3.2 `void __gnu_debug::Safe_unordered_container_base::M_attach_local (_Safe_iterator_base * __it, bool __constant)`

Attach an iterator to this container.

4.88.3.3 `void __gnu_debug::Safe_unordered_container_base::M_attach_local.single (_Safe_iterator_base * __it, bool __constant) throw ()`

Likewise but not thread safe.

4.88.3.4 `void __gnu_debug::Safe_sequence_base::M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
`[inherited]`

Likewise but not thread safe.

4.88.3.5 `void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it)` `[inherited]`

Detach an iterator from this sequence

4.88.3.6 `void __gnu_debug::Safe_unordered_container_base::M_detach_all ()` `[protected]`

Detach all iterators, leaving them singular.

4.88.3.7 `void __gnu_debug::Safe_unordered_container_base::M_detach_local (_Safe_iterator_base * __it)`

Detach an iterator from this container

4.88.3.8 `void __gnu_debug::Safe_unordered_container_base::M_detach_local_single (_Safe_iterator_base * __it) throw ()`

Likewise but not thread safe.

4.88.3.9 `void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw ()` [inherited]

Likewise but not thread safe.

4.88.3.10 `void __gnu_debug::Safe_sequence_base::M_detach_singular ()` [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

4.88.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw ()` [protected], [inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.88.3.12 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const` [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

References `__gnu_debug::Safe_sequence_base::M_version`.

4.88.3.13 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.88.3.14 `void __gnu_debug::Safe_unordered_container_base::M_swap (_Safe_unordered_container_base & __x)` [protected]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.88.3.15 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x)` [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.88.4 Member Data Documentation

4.88.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.88.4.2 __Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M_const_local_iterators

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.88.4.3 __Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if().

4.88.4.4 __Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M_local_iterators

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.88.4.5 unsigned int __gnu_debug::Safe_sequence_base::M_version [mutable],[inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

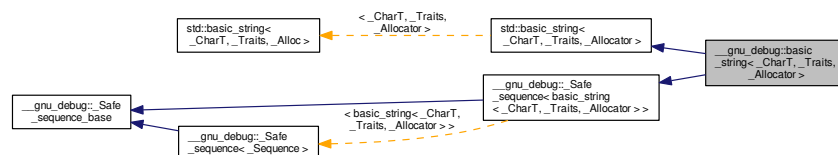
Referenced by __gnu_debug::Safe_sequence_base::M_invalidate_all(), and __gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if().

The documentation for this class was generated from the following file:

- [safe_unordered_base.h](#)

4.89 __gnu_debug::basic_string<_CharT, _Traits, _Allocator> Class Template Reference

Inheritance diagram for __gnu_debug::basic_string<_CharT, _Traits, _Allocator>:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator`
`< typename`
`_Base::const_iterator,`
`basic_string >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**

- typedef `std::reverse_iterator`
 < `const_iterator` > **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef
 `__gnu_debug::Safe_iterator`
 < typename `_Base::iterator`,
 `basic_string` > **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator`
 < `iterator` > **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits::char_type` **value_type**

Public Member Functions

- **basic_string** (const `_Allocator` & __a= `_Allocator`())
- **basic_string** (const `_Base` & __base)
- **basic_string** (const `basic_string` & __str)
- **basic_string** (const `basic_string` & __str, `size_type` __pos, `size_type` __n= `_Base::npos`, const `_Allocator` & __a= `_Allocator`())
- **basic_string** (const `_CharT` * __s, `size_type` __n, const `_Allocator` & __a= `_Allocator`())
- **basic_string** (const `_CharT` * __s, const `_Allocator` & __a= `_Allocator`())
- **basic_string** (`size_type` __n, `_CharT` __c, const `_Allocator` & __a= `_Allocator`())
- template<typename `_InputIterator` >
 basic_string (`_InputIterator` __begin, `_InputIterator` __end, const `_Allocator` & __a= `_Allocator`())
- **basic_string** (`basic_string` && __str) noexcept
- **basic_string** (`std::initializer_list`< `_CharT` > __l, const `_Allocator` & __a= `_Allocator`())
- void `_M_attach` (`_Safe_iterator_base` * __it, bool __constant)
- void `_M_attach_single` (`_Safe_iterator_base` * __it, bool __constant) throw ()
- `_Base` & `_M_base` () noexcept
- const `_Base` & `_M_base` () const noexcept
- void `_M_detach` (`_Safe_iterator_base` * __it)
- void `_M_detach_single` (`_Safe_iterator_base` * __it) throw ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` __pred)
- void `_M_transfer_from_if` (`_Safe_sequence` & __from, `_Predicate` __pred)
- `basic_string` & **append** (const `basic_string` & __str)
- `basic_string` & **append** (const `basic_string` & __str, `size_type` __pos, `size_type` __n)
- `basic_string` & **append** (const `_CharT` * __s, `size_type` __n)
- `basic_string` & **append** (const `_CharT` * __s)
- `basic_string` & **append** (`size_type` __n, `_CharT` __c)
- template<typename `_InputIterator` >
 basic_string & **append** (`_InputIterator` __first, `_InputIterator` __last)
- `basic_string` & **append** (const `basic_string` & __str)
- `basic_string` & **append** (const `basic_string` & __str, `size_type` __pos, `size_type` __n)
- `basic_string` & **append** (`initializer_list`< `_CharT` > __l)
- `basic_string` & **assign** (const `basic_string` & __x)
- `basic_string` & **assign** (`basic_string` && __x)
- `basic_string` & **assign** (const `basic_string` & __str, `size_type` __pos, `size_type` __n)

- `basic_string` & **assign** (const `_CharT` * __s, size_type __n)
- `basic_string` & **assign** (const `_CharT` * __s)
- `basic_string` & **assign** (size_type __n, `_CharT` __c)
- template<typename `_InputIterator` >
`basic_string` & **assign** (`_InputIterator` __first, `_InputIterator` __last)
- `basic_string` & **assign** (std::initializer_list< `_CharT` > __l)
- `basic_string` & **assign** (const `basic_string` & __str)
- `basic_string` & **assign** (`basic_string` && __str)
- `basic_string` & **assign** (const `basic_string` & __str, size_type __pos, size_type __n)
- const_reference **at** (size_type __n) const
- reference **at** (size_type __n)
- reference **back** ()
- const_reference **back** () const
- **iterator begin** () noexcept
- **const_iterator begin** () const noexcept
- const `_CharT` * **c_str** () const noexcept
- size_type **capacity** () const noexcept
- **const_iterator cbegin** () const noexcept
- **const_iterator cend** () const noexcept
- void **clear** () noexcept
- int **compare** (const `basic_string` & __str) const
- int **compare** (size_type __pos1, size_type __n1, const `basic_string` & __str) const
- int **compare** (size_type __pos1, size_type __n1, const `basic_string` & __str, size_type __pos2, size_type __n2) const
- int **compare** (const `_CharT` * __s) const
- int **compare** (size_type __pos1, size_type __n1, const `_CharT` * __s) const
- int **compare** (size_type __pos1, size_type __n1, const `_CharT` * __s, size_type __n2) const
- int **compare** (const `basic_string` & __str) const
- int **compare** (size_type __pos, size_type __n, const `basic_string` & __str) const
- int **compare** (size_type __pos1, size_type __n1, const `basic_string` & __str, size_type __pos2, size_type __n2) const
- size_type **copy** (`_CharT` * __s, size_type __n, size_type __pos=0) const
- **const_reverse_iterator crbegin** () const noexcept
- **const_reverse_iterator crend** () const noexcept
- const `_CharT` * **data** () const noexcept
- bool **empty** () const noexcept
- **iterator end** () noexcept
- **const_iterator end** () const noexcept
- `basic_string` & **erase** (size_type __pos=0, size_type __n= `_Base::npos`)
- **iterator erase** (**iterator** __position)
- **iterator erase** (**iterator** __first, **iterator** __last)
- **iterator erase** (**iterator** __position)
- **iterator erase** (**iterator** __first, **iterator** __last)
- size_type **find** (const `basic_string` & __str, size_type __pos=0) const noexcept
- size_type **find** (const `_CharT` * __s, size_type __pos, size_type __n) const
- size_type **find** (const `_CharT` * __s, size_type __pos=0) const
- size_type **find** (`_CharT` __c, size_type __pos=0) const noexcept
- size_type **find** (const `basic_string` & __str, size_type __pos=0) const noexcept
- size_type **find_first_not_of** (const `basic_string` & __str, size_type __pos=0) const noexcept
- size_type **find_first_not_of** (const `_CharT` * __s, size_type __pos, size_type __n) const
- size_type **find_first_not_of** (const `_CharT` * __s, size_type __pos=0) const

- `size_type find_first_not_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_not_of` (`const basic_string &__str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const basic_string &__str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const basic_string &__str`, `size_type __pos=0`) `const noexcept`
- `size_type find_last_not_of` (`const basic_string &__str`, `size_type __pos=_Base::npos`) `const noexcept`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=_Base::npos`) `const noexcept`
- `size_type find_last_not_of` (`const basic_string &__str`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_of` (`const basic_string &__str`, `size_type __pos=_Base::npos`) `const noexcept`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos=_Base::npos`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=_Base::npos`) `const noexcept`
- `size_type find_last_of` (`const basic_string &__str`, `size_type __pos=npos`) `const noexcept`
- reference `front` ()
- `const_reference front` () `const`
- `allocator_type get_allocator` () `const noexcept`
- `basic_string &insert` (`size_type __pos1`, `const basic_string &__str`)
- `basic_string &insert` (`size_type __pos1`, `const basic_string &__str`, `size_type __pos2`, `size_type __n`)
- `basic_string &insert` (`size_type __pos`, `const _CharT *__s`, `size_type __n`)
- `basic_string &insert` (`size_type __pos`, `const _CharT *__s`)
- `basic_string &insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `iterator insert` (`iterator __p`, `_CharT __c`)
- `void insert` (`iterator __p`, `size_type __n`, `_CharT __c`)
- `template<typename _InputIterator>`
`void insert` (`iterator __p`, `_InputIterator __first`, `_InputIterator __last`)
- `void insert` (`iterator __p`, `std::initializer_list<_CharT> __l`)
- `void insert` (`iterator __p`, `size_type __n`, `_CharT __c`)
- `void insert` (`iterator __p`, `_InputIterator __beg`, `_InputIterator __end`)
- `void insert` (`iterator __p`, `initializer_list<_CharT> __l`)
- `basic_string &insert` (`size_type __pos1`, `const basic_string &__str`)
- `basic_string &insert` (`size_type __pos1`, `const basic_string &__str`, `size_type __pos2`, `size_type __n`)
- `iterator insert` (`iterator __p`, `_CharT __c`)
- `size_type length` () `const noexcept`
- `size_type max_size` () `const noexcept`
- `basic_string &operator+=` (`const basic_string &__str`)
- `basic_string &operator+=` (`const _CharT *__s`)
- `basic_string &operator+=` (`_CharT __c`)
- `basic_string &operator+=` (`std::initializer_list<_CharT> __l`)
- `basic_string &operator+=` (`const basic_string &__str`)
- `basic_string &operator=` (`const basic_string &__str`)
- `basic_string &operator=` (`const _CharT *__s`)
- `basic_string &operator=` (`_CharT __c`)
- `basic_string &operator=` (`basic_string &&__str`)
- `basic_string &operator=` (`std::initializer_list<_CharT> __l`)
- `const_reference operator[]` (`size_type __pos`) `const`
- reference `operator[]` (`size_type __pos`)
- `void pop_back` ()

- void **push_back** (`_CharT __c`)
- `reverse_iterator` **rbegin** () noexcept
- `const_reverse_iterator` **rbegin** () const noexcept
- `reverse_iterator` **rend** () noexcept
- `const_reverse_iterator` **rend** () const noexcept
- `basic_string` & **replace** (`size_type __pos1`, `size_type __n1`, const `basic_string` & `__str`)
- `basic_string` & **replace** (`size_type __pos1`, `size_type __n1`, const `basic_string` & `__str`, `size_type __pos2`, `size_type __n2`)
- `basic_string` & **replace** (`size_type __pos`, `size_type __n1`, const `_CharT *` `__s`, `size_type __n2`)
- `basic_string` & **replace** (`size_type __pos`, `size_type __n1`, const `_CharT *` `__s`)
- `basic_string` & **replace** (`size_type __pos`, `size_type __n1`, `size_type __n2`, `_CharT __c`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `basic_string` & `__str`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `_CharT *` `__s`, `size_type __n`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `_CharT *` `__s`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `size_type __n`, `_CharT __c`)
- `template<typename _InputIterator>`
`basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `_InputIterator __j1`, `_InputIterator __j2`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `std::initializer_list<_CharT> __l`)
- `basic_string` & **replace** (`size_type __pos`, `size_type __n`, const `basic_string` & `__str`)
- `basic_string` & **replace** (`size_type __pos1`, `size_type __n1`, const `basic_string` & `__str`, `size_type __pos2`, `size_type __n2`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `basic_string` & `__str`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `_CharT *` `__s`, `size_type __n`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `_CharT *` `__s`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `size_type __n`, `_CharT __c`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `_InputIterator __k1`, `_InputIterator __k2`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `_CharT *` `__k1`, `_CharT *` `__k2`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, const `_CharT *` `__k1`, const `_CharT *` `__k2`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `iterator __k1`, `iterator __k2`)
- `basic_string` & **replace** (`iterator __i1`, `iterator __i2`, `initializer_list<_CharT> __l`)
- void **reserve** (`size_type __res_arg=0`)
- void **resize** (`size_type __n`, `_CharT __c`)
- void **resize** (`size_type __n`)
- `size_type` **rfind** (const `basic_string` & `__str`, `size_type __pos=_Base::npos`) const noexcept
- `size_type` **rfind** (const `_CharT *` `__s`, `size_type __pos`, `size_type __n`) const
- `size_type` **rfind** (const `_CharT *` `__s`, `size_type __pos=_Base::npos`) const
- `size_type` **rfind** (`_CharT __c`, `size_type __pos=_Base::npos`) const noexcept
- `size_type` **rfind** (const `basic_string` & `__str`, `size_type __pos=npos`) const noexcept
- void **shrink_to_fit** ()
- `size_type` **size** () const noexcept
- `basic_string` **substr** (`size_type __pos=0`, `size_type __n=_Base::npos`) const
- void **swap** (`basic_string<_CharT, _Traits, _Allocator> & __x`)
- void **swap** (`basic_string` & `__s`)

Public Attributes

- `_Safe_iterator_base *` `_M_const_iterators`
- `_Safe_iterator_base *` `_M_iterators`
- unsigned int `_M_version`

Static Public Attributes

- static const size_type `npos`

Protected Member Functions

- void `_M_detach_all()`
- void `_M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex()` throw ()
- void `_M_revalidate_singular()`
- void `_M_swap(_Safe_sequence_base &__x)`

4.89.1 Detailed Description

template<typename `_CharT`, typename `_Traits` = `std::char_traits<_CharT>`, typename `_Allocator` = `std::allocator<_CharT>`>>class `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>`

Class `std::basic_string` with safety/checking/debug instrumentation.

Definition at line 41 of file `debug/string`.

4.89.2 Member Function Documentation

4.89.2.1 void `__gnu_debug::Safe_sequence_base::M.attach(_Safe_iterator_base * __it, bool __constant)` [inherited]

Attach an iterator to this sequence.

4.89.2.2 void `__gnu_debug::Safe_sequence_base::M.attach_single(_Safe_iterator_base * __it, bool __constant)` throw () [inherited]

Likewise but not thread safe.

4.89.2.3 void `__gnu_debug::Safe_sequence_base::M.detach(_Safe_iterator_base * __it)` [inherited]

Detach an iterator from this sequence

4.89.2.4 void `__gnu_debug::Safe_sequence_base::M.detach_all()` [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()`.

4.89.2.5 void `__gnu_debug::Safe_sequence_base::M.detach_single(_Safe_iterator_base * __it)` throw () [inherited]

Likewise but not thread safe.

4.89.2.6 void `__gnu_debug::Safe_sequence_base::M.detach_singular()` [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

4.89.2.7 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw ()` [protected], [inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.89.2.8 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const` [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::M_version`.

4.89.2.9 `void __gnu_debug::Safe_sequence<basic_string<_CharT, _Traits, _Allocator>>::M_invalidate_if (_Predicate __pred)` [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.89.2.10 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.89.2.11 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x)` [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.89.2.12 `void __gnu_debug::Safe_sequence<basic_string<_CharT, _Traits, _Allocator>>::M_transfer_from_if (_Safe_sequence<basic_string<_CharT, _Traits, _Allocator>> & __from, _Predicate __pred)` [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.89.2.13 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const basic_string<_CharT, _Traits, _Allocator> & __str)` [inherited]

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

4.89.2.14 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n)` [inherited]

Append a substring.

Parameters

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

4.89.2.15 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (initializer_list<_CharT> __l)`
`[inline], [inherited]`

Append an `initializer_list` of characters.

Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to append.
------------------	--

Returns

Reference to this string.

Definition at line 1030 of file `basic_string.h`.

4.89.2.16 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const basic_string<_CharT, _Traits, _Allocator> &__str)` `[inherited]`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

4.89.2.17 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (basic_string<_CharT, _Traits, _Allocator> &&__str)` `[inline], [inherited]`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 1079 of file `basic_string.h`.

4.89.2.18 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n)` `[inline]`, `[inherited]`

Set value to a substring of a string.

Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of <code>str</code> .
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 1100 of file `basic_string.h`.

4.89.2.19 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::at (size_type __n) const` `[inline]`, `[inherited]`

Provides access to the data contained in the string.

Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

Returns

Read-only (const) reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 864 of file `basic_string.h`.

4.89.2.20 `reference std::basic_string<_CharT, _Traits, _Allocator>::at (size_type __n)` `[inline]`, `[inherited]`

Provides access to the data contained in the string.

Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

Returns

Read/write reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 883 of file `basic_string.h`.

4.89.2.21 `reference std::basic_string<_CharT, _Traits, _Allocator>::back ()` `[inline]`, `[inherited]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 913 of file `basic_string.h`.

4.89.2.22 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::back () const` `[inline]`, `[inherited]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 921 of file `basic_string.h`.

4.89.2.23 `size_type std::basic_string<_CharT, _Traits, _Allocator>::capacity () const` `[inline]`, `[noexcept]`, `[inherited]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 776 of file `basic_string.h`.

4.89.2.24 `int std::basic_string<_CharT, _Traits, _Allocator>::compare (const basic_string<_CharT, _Traits, _Allocator> & __str) const` `[inline]`, `[inherited]`

Compare to a string.

Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2225 of file basic_string.h.

4.89.225 `int std::basic_string<_CharT, _Traits, _Allocator >::compare (size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Allocator > & __str) const` `[inherited]`

Compare substring to a string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.89.226 `int std::basic_string<_CharT, _Traits, _Allocator >::compare (size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator > & __str, size_type __pos2, size_type __n2) const` `[inherited]`

Compare substring to a substring.

Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of str.
<code>__n2</code>	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.89.227 `bool std::basic_string<_CharT, _Traits, _Allocator >::empty () const` `[inline],[noexcept],`
`[inherited]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 811 of file basic_string.h.

4.89.2.28 `iterator std::basic_string<_CharT, _Traits, _Allocator>::erase (iterator __position)` `[inline]`,
`[inherited]`

Remove one character.

Parameters

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1378 of file `basic_string.h`.

4.89.2.29 `iterator std::basic_string<_CharT, _Traits, _Allocator>::erase (iterator __first, iterator __last)` `[inherited]`

Remove a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

4.89.2.30 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0) const` `[inline]`, `[noexcept]`, `[inherited]`

Find position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1846 of file `basic_string.h`.

4.89.2.31 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0) const` `[inline]`, `[noexcept]`, `[inherited]`

Find position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2079 of file `basic_string.h`.

4.89.2.32 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of(const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0) const` `[inline]`, `[noexcept]`, `[inherited]`

Find position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1952 of file `basic_string.h`.

4.89.2.33 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of(const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos) const` `[inline]`, `[noexcept]`, `[inherited]`

Find last position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2142 of file `basic_string.h`.

4.89.2.34 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of(const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos) const` `[inline]`, `[noexcept]`, `[inherited]`

Find last position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2016 of file `basic_string.h`.

4.89.2.35 `reference std::basic_string<_CharT, _Traits, _Allocator>::front ()` `[inline]`, `[inherited]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 897 of file `basic_string.h`.

4.89.2.36 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::front () const` `[inline]`, `[inherited]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 905 of file `basic_string.h`.

4.89.2.37 `allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator () const` `[inline]`, `[noexcept]`, `[inherited]`

Return copy of allocator used to construct this string.

Definition at line 1817 of file `basic_string.h`.

4.89.2.38 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, size_type __n, _CharT __c)` `[inline]`, `[inherited]`

Insert multiple characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1185 of file `basic_string.h`.

4.89.2.39 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, InputIterator __beg, InputIterator __end)` `[inline]`, `[inherited]`

Insert a range of characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1202 of file `basic_string.h`.

4.89.2.40 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, initializer_list<_CharT> __l)`
`[inline], [inherited]`

Insert an `initializer_list` of characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 1213 of file `basic_string.h`.

4.89.2.41 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str)` `[inline], [inherited]`

Insert value of a string.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file `basic_string.h`.

4.89.2.42 **basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n)** [inline], [inherited]

Insert a substring.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in str to insert.
<code>__n</code>	Number of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>pos1 > size()</code> or <code>__pos2 > str.size()</code> .

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1255 of file `basic_string.h`.

4.89.2.43 **iterator std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, _CharT __c)** [inline], [inherited]

Insert one character.

Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1337 of file `basic_string.h`.

4.89.2.44 **size_type std::basic_string<_CharT, _Traits, _Allocator>::length () const** [inline], [noexcept], [inherited]

Returns the number of characters in the string, not including any null-termination.

Definition at line 721 of file basic_string.h.

4.89.2.45 `size_type std::basic_string<_CharT, _Traits, _Allocator >::max_size () const` `[inline], [noexcept], [inherited]`

Returns the size() of the largest possible string.

Definition at line 726 of file basic_string.h.

4.89.2.46 `basic_string& std::basic_string<_CharT, _Traits, _Allocator >::operator+=(const basic_string<_CharT, _Traits, _Allocator > &__str)` `[inline], [inherited]`

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 932 of file basic_string.h.

4.89.2.47 `basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Allocator > &__str)` `[inline], [inherited]`

Replace characters with value from another string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1429 of file basic_string.h.

4.89.2.48 `basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator > &__str, size_type __pos2, size_type __n2)` `[inline], [inherited]`

Replace characters with value from another string.

Parameters

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of str to use.
<code>__n2</code>	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos1 > size()</code> or <code>__pos2 > __str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1451 of file `basic_string.h`.

4.89.2.49 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, const basic_string<_CharT, _Traits, _Allocator> & __str)` `[inline]`, `[inherited]`

Replace range of characters with string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file `basic_string.h`.

4.89.2.50 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)` `[inline]`, `[inherited]`

Replace range of characters with C substring.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1557 of file `basic_string.h`.

4.89.251 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, const _CharT * __s)` `[inline]`, `[inherited]`

Replace range of characters with C string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1578 of file `basic_string.h`.

4.89.252 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)` `[inline]`, `[inherited]`

Replace range of characters with multiple characters.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1599 of file `basic_string.h`.

4.89.2.53 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)` `[inline]`, `[inherited]`

Replace range of characters with range.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1623 of file `basic_string.h`.

4.89.2.54 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, initializer_list<_CharT> __l)` `[inline]`, `[inherited]`

Replace range of characters with `initializer_list`.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1692 of file `basic_string.h`.

4.89.2.55 `void std::basic_string<_CharT, _Traits, _Allocator>::reserve (size_type __res_arg = 0)` `[inherited]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

Exceptions

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

4.89.2.56 `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos) const` `[inline]`, `[noexcept]`, `[inherited]`

Find last position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1891 of file `basic_string.h`.

4.89.2.57 `size_type std::basic_string<_CharT, _Traits, _Allocator>::size () const` `[inline]`, `[noexcept]`, `[inherited]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 715 of file `basic_string.h`.

4.89.2.58 `void std::basic_string<_CharT, _Traits, _Allocator>::swap (basic_string<_CharT, _Traits, _Allocator> & __s)` `[inherited]`

Swap contents with another string.

Parameters

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

4.89.3 Member Data Documentation

4.89.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.89.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.89.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.89.3.4 `const size_type std::basic_string<_CharT, _Traits, _Allocator>::npos` [static], [inherited]

Value returned by various member functions when they fail.

Definition at line 285 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

4.90 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp` & `__b`)
- `template<typename _Result, typename _Addend> _Result operator() (const _Result &__x, const _Addend &__y)`

Public Attributes

- `_BinOp` & `__binop`

4.90.1 Detailed Description

```
template<typename _BinOp>struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>
```

General reduction, using a binary operator.

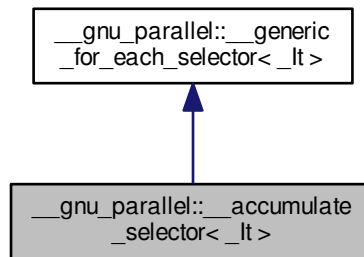
Definition at line 335 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.91 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`std::iterator_traits<_It>`
`::value_type operator() (_Op __o, _It __i)`

Public Attributes

- `_It` [_M_finish_iterator](#)

4.91.1 Detailed Description

```
template<typename _It>struct __gnu_parallel::__accumulate_selector<_It>
```

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

4.91.2 Member Function Documentation

4.91.2.1 `template<typename _It> template<typename _Op> std::iterator_traits<_It>::value_type
__gnu_parallel::__accumulate_selector<_It>::operator()(_Op __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator (unused).
<code>__i</code>	iterator referencing object.

Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

4.91.3 Member Data Documentation

4.91.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

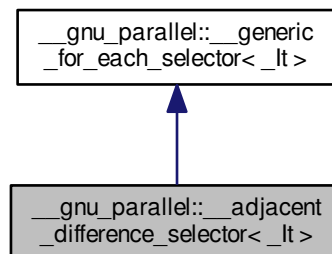
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.92 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



Public Member Functions

- `template<typename _Op >`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.92.1 Detailed Description

`template<typename _It>struct __gnu_parallel::__adjacent_difference_selector< _It >`

Selector that returns the difference between two adjacent __elements.

Definition at line 269 of file `for_each_selectors.h`.

4.92.2 Member Data Documentation

4.92.2.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::__M_finish_iterator`
[inherited]

__Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

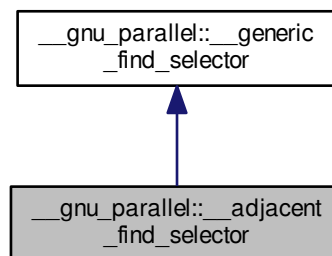
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- `for_each_selectors.h`

4.93 __gnu_parallel::__adjacent_find_selector Struct Reference

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair<_RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

4.93.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

4.93.2 Member Function Documentation

4.93.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__adjacent_find_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,`
 `_Pred __pred) \[inline\]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 105 of file `find_selectors.h`.

References `std::make_pair()`.

4.93.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__adjacent_find_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) \[inline\]`

Test on one position.

Parameters

<code>__i1</code>	Iterator on first sequence.
<code>__i2</code>	Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

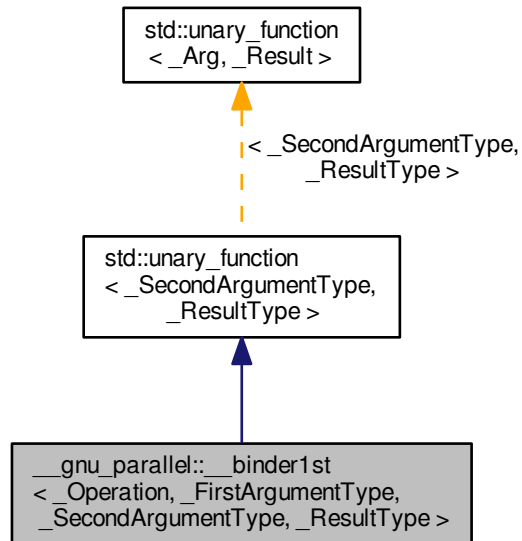
Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.94 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



Public Types

- typedef `_SecondArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- **`binder1st`** (const `_Operation` &__x, const `_FirstArgumentType` &__y)
- `_ResultType` **`operator()`** (const `_SecondArgumentType` &__x)
- `_ResultType` **`operator()`** (`_SecondArgumentType` &__x) const

Protected Attributes

- `_Operation` **`_M_op`**
- `_FirstArgumentType` **`_M_value`**

4.94.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>class __-  
gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>
```

Similar to `std::binder1st`, but giving the argument types explicitly.

Definition at line 192 of file `parallel/base.h`.

4.94.2 Member Typedef Documentation

4.94.2.1 `typedef _SecondArgumentType std::unary_function<_SecondArgumentType, _ResultType>::argument_type`
[inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.94.2.2 `typedef _ResultType std::unary_function<_SecondArgumentType, _ResultType>::result_type` [inherited]

`result_type` is the return type

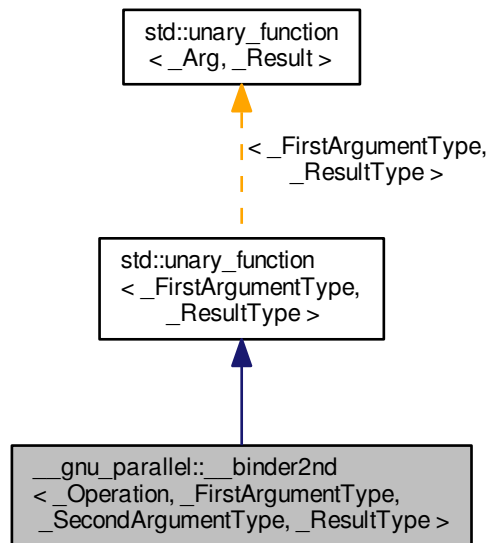
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.95 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



Public Types

- typedef `_FirstArgumentType` `argument_type`
- typedef `_ResultType` `result_type`

Public Member Functions

- **`__binder2nd`** (`const _Operation &__x`, `const _SecondArgumentType &__y`)
- `_ResultType` **`operator()`** (`const _FirstArgumentType &__x`) `const`
- `_ResultType` **`operator()`** (`_FirstArgumentType &__x`)

Protected Attributes

- `_Operation` **`_M_op`**
- `_SecondArgumentType` **`_M_value`**

4.95.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>class __-
gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder2nd`, but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

4.95.2 Member Typedef Documentation

4.95.2.1 `typedef _FirstArgumentType std::unary_function< _FirstArgumentType , _ResultType >::argument_type`
[inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.95.2.2 `typedef _ResultType std::unary_function< _FirstArgumentType , _ResultType >::result_type` [inherited]

`result_type` is the return type

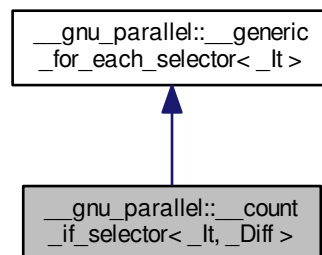
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.96 __gnu_parallel::__count_if_selector<_It, _Diff> Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_if_selector<_It, _Diff>`:



Public Member Functions

- `template<typename _Op >`
`_Diff operator() (_Op &__o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

4.96.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_if_selector< _It, _Diff >`

`std::count_if()` selector.

Definition at line 194 of file `for_each_selectors.h`.

4.96.2 Member Function Documentation

4.96.2.1 `template<typename _It, typename _Diff> template<typename _Op> _Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator()(_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

4.96.3 Member Data Documentation

4.96.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

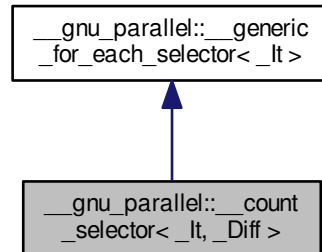
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.97 `__gnu_parallel::__count_selector< _It, _Diff >` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_selector< _It, _Diff >`:



Public Member Functions

- `template<typename _ValueType > _Diff operator() (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.97.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_selector< _It, _Diff >`

`std::count()` selector.

Definition at line 180 of file `for_each_selectors.h`.

4.97.2 Member Function Documentation

4.97.2.1 `template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (_ValueType & __v, _It __i) [inline]`

Functor execution.

Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

4.97.3 Member Data Documentation**4.97.3.1** `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator`
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

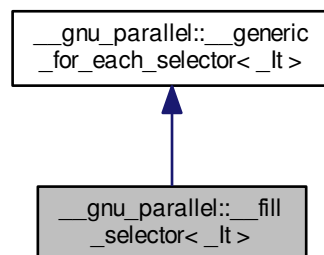
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.98 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:

**Public Member Functions**

- `template<typename _ValueType>`
`bool operator() (_ValueType &__v, _It __i)`

Public Attributes

- `_It` [M_finish_iterator](#)

4.98.1 Detailed Description

template<typename _It>struct __gnu_parallel::__fill_selector<_It>

std::fill() selector.

Definition at line 84 of file for_each_selectors.h.

4.98.2 Member Function Documentation

4.98.2.1 template<typename _It> template<typename _ValueType> bool __gnu_parallel::__fill_selector<_It>::operator()
(_ValueType & __v, _It __i) [inline]

Functor execution.

Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Definition at line 91 of file for_each_selectors.h.

4.98.3 Member Data Documentation

4.98.3.1 template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator
[inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

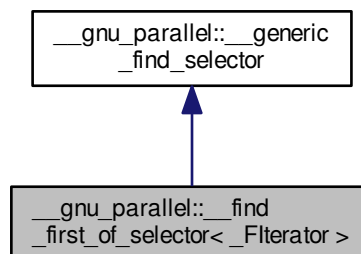
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.99 __gnu_parallel::__find_first_of_selector<_FIterator> Struct Template Reference

Inheritance diagram for __gnu_parallel::__find_first_of_selector<_FIterator>:



Public Member Functions

- **__find_first_of_selector** (*_FIterator* __begin, *_FIterator* __end)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`std::pair<_RAIter1, _RAIter2> __M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

Public Attributes

- *_FIterator* **_M_begin**
- *_FIterator* **_M_end**

4.99.1 Detailed Description

`template<typename _FIterator>struct __gnu_parallel::__find_first_of_selector<_FIterator>`

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

4.99.2 Member Function Documentation

4.99.2.1 `template<typename _FIterator> template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)` `[inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 186 of file `find_selectors.h`.

References `std::make_pair()`.

4.99.2.2 `template<typename _FIterator> template<typename _RAIter1, typename _RAIter2, typename _Pred> bool`
`__gnu_parallel::__find_first_of_selector<_FIterator>::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`
`[inline]`

Test on one position.

Parameters

<code>__i1</code>	<i>_Iterator</i> on first sequence.
<code>__i2</code>	<i>_Iterator</i> on second sequence (unused).
<code>__pred</code>	Find predicate.

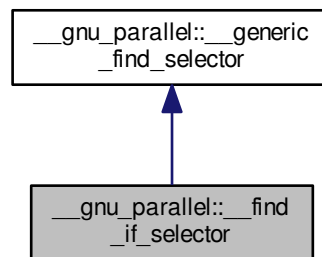
Definition at line 169 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.100 `__gnu_parallel::__find_if_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

4.100.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

4.100.2 Member Function Documentation

- 4.100.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__find_if_selector::_M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

```
4.100.2.2  template<typename _RAIter1, typename _RAIter2, typename _Pred> bool __gnu_parallel::__find_if_selector::operator() (
    _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]
```

Test on one position.

Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

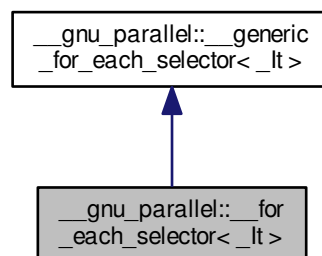
Definition at line 60 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.101 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

4.101.1 Detailed Description

`template<typename _It>struct __gnu_parallel::__for_each_selector<_It>`

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

4.101.2 Member Function Documentation

4.101.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__for_each_selector<_It>::operator()
(_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

4.101.3 Member Data Documentation

4.101.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator
[inherited]`

`_It` iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

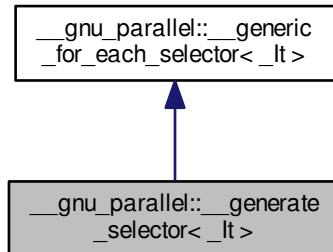
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.102 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.102.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector<_It>`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

4.102.2 Member Function Documentation

4.102.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__generate_selector<_It>::operator() (_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

4.102.3 Member Data Documentation

4.102.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator`
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

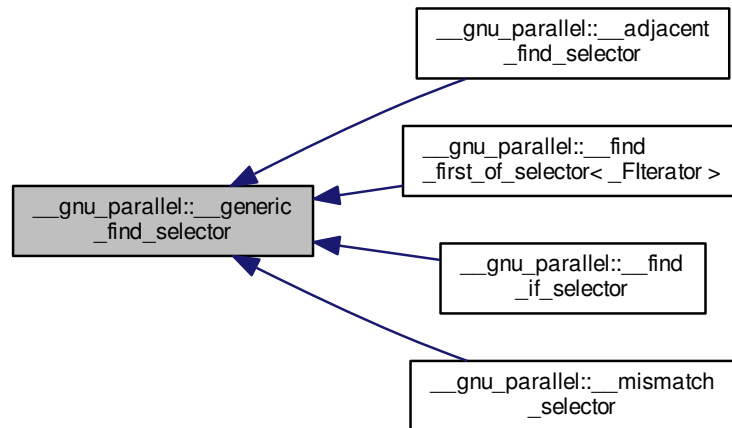
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.103 `__gnu_parallel::__generic_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



4.103.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

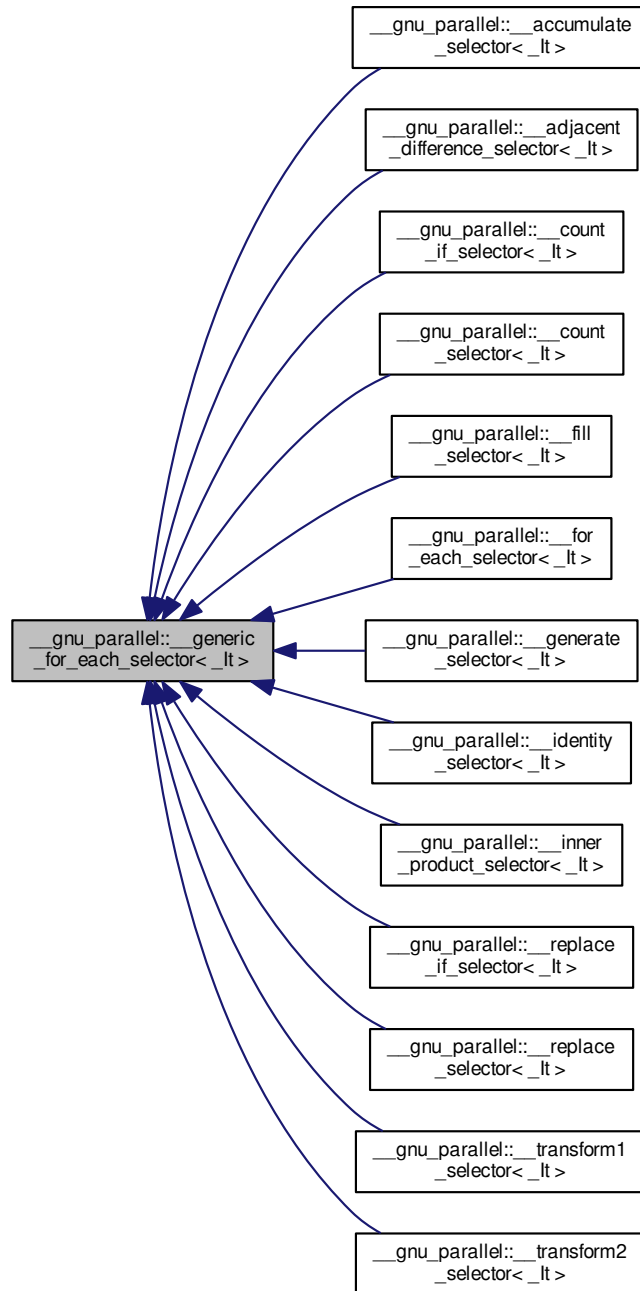
Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.104 __gnu_parallel::__generic_for_each_selector<_It> Struct Template Reference

Inheritance diagram for __gnu_parallel::__generic_for_each_selector<_It>:



Public Attributes

- [_It _M_finish_iterator](#)

4.104.1 Detailed Description

`template<typename _It>struct __gnu_parallel::__generic_for_each_selector<_It>`

Generic `__selector` for embarrassingly parallel functions.

Definition at line 42 of file `for_each_selectors.h`.

4.104.2 Member Data Documentation

4.104.2.1 `template<typename _It>_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

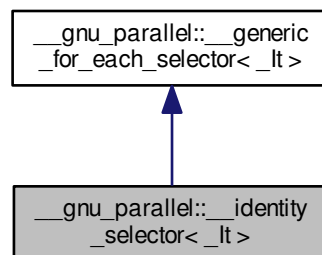
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.105 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__identity_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`_It operator() (_Op __o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

4.105.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__identity_selector<_It>`

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

4.105.2 Member Function Documentation

4.105.2.1 `template<typename _It> template<typename _Op> _It __gnu_parallel::__identity_selector<_It>::operator() (_Op __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator (unused).
<code>__i</code>	iterator referencing object.

Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

4.105.3 Member Data Documentation

4.105.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

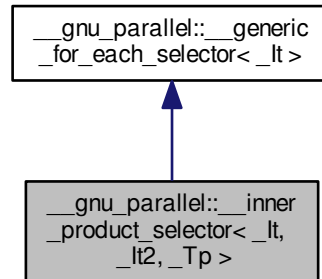
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.106 `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`:



Public Member Functions

- [__inner_product_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op >`
`_Tp operator()` (`_Op __mult, _It __current`)

Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

4.106.1 Detailed Description

`template<typename _It, typename _It2, typename _Tp> struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`

`std::inner_product()` selector.

Definition at line 222 of file `for_each_selectors.h`.

4.106.2 Constructor & Destructor Documentation

4.106.2.1 `template<typename _It, typename _It2, typename _Tp> __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__inner_product_selector (_It __b1, _It2 __b2)` [`inline`], [`explicit`]

Constructor.

Parameters

<code>__b1</code>	Begin iterator of first sequence.
<code>__b2</code>	Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

4.106.3 Member Function Documentation

4.106.3.1 `template<typename _It, typename _It2, typename _Tp> template<typename _Op> _Tp
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()(_Op __mult, _It __current)
[inline]`

Functor execution.

Parameters

<code>__mult</code>	Multiplication functor.
<code>__current</code>	iterator referencing object.

Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_iterator`, and `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin2_iterator`.

4.106.4 Member Data Documentation

4.106.4.1 `template<typename _It, typename _It2, typename _Tp> _It __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_iterator`

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator>()()`.

4.106.4.2 `template<typename _It, typename _It2, typename _Tp> _It2 __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator>()()`.

4.106.4.3 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.107 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

Public Attributes

- `_Compare & __comp`

4.107.1 Detailed Description

`template<typename _Compare, typename _It>struct __gnu_parallel::__max_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.108 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

Public Attributes

- `_Compare & __comp`

4.108.1 Detailed Description

`template<typename _Compare, typename _It>struct __gnu_parallel::__min_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

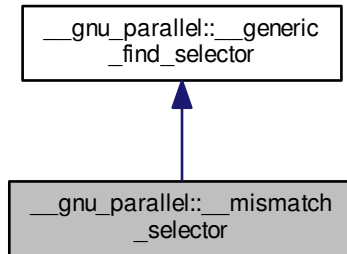
Definition at line 307 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.109 __gnu_parallel::__mismatch_selector Struct Reference

Inheritance diagram for __gnu_parallel::__mismatch_selector:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair<_RAIter1, _RAIter2> _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

4.109.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

4.109.2 Member Function Documentation

4.109.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__mismatch_selector::_M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,`
`_Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 143 of file `find_selectors.h`.

4.109.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.110 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

4.110.1 Detailed Description

`template<bool __sentinels, typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned off.

Note that 3-way merging is always stable!

Definition at line 752 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.111 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

4.111.1 Detailed Description

`template<typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned on.

Note that 3-way merging is always stable!

Definition at line 772 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.112 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

4.112.1 Detailed Description

`template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned off.

Note that 4-way merging is always stable!

Definition at line 795 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.113 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

4.113.1 Detailed Description

`template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

Definition at line 815 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.114 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterliterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

4.114.1 Detailed Description

`template<bool __sentinels, bool __stable, typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with `__sentinels` turned on.

Definition at line 837 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.115 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterliterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

4.115.1 Detailed Description

`template<bool __stable, typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with `__sentinels` turned off.

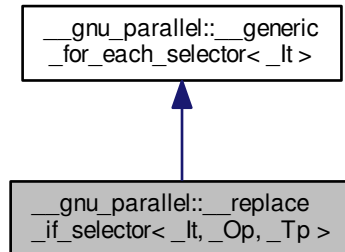
Definition at line 872 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.116 `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



Public Member Functions

- `__replace_if_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Op` & `__o`, `_It` `__i`)

Public Attributes

- const `_Tp` & `__new_val`
- `_It` `_M_finish_iterator`

4.116.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp> struct __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>
```

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

4.116.2 Constructor & Destructor Documentation

4.116.2.1 `template<typename _It, typename _Op, typename _Tp> __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__replace_if_selector (const _Tp & __new_val) [inline], [explicit]`

Constructor.

Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 164 of file `for_each_selectors.h`.

4.116.3 Member Function Documentation

4.116.3.1 `template<typename _It, typename _Op, typename _Tp> bool __gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::operator()(_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::__new_val`.

4.116.4 Member Data Documentation

4.116.4.1 `template<typename _It, typename _Op, typename _Tp> const _Tp& __gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::operator()()`.

4.116.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

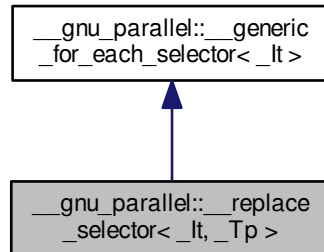
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.117 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



Public Member Functions

- `__replace_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Tp` & `__v`, `_It` `__i`)

Public Attributes

- const `_Tp` & `__new_val`
- `_It` `_M_finish_iterator`

4.117.1 Detailed Description

```
template<typename _It, typename _Tp> struct __gnu_parallel::__replace_selector<_It, _Tp>
```

`std::replace()` selector.

Definition at line 132 of file `for_each_selectors.h`.

4.117.2 Constructor & Destructor Documentation

4.117.2.1 `template<typename _It, typename _Tp> __gnu_parallel::__replace_selector<_It, _Tp>::__replace_selector (const _Tp & __new_val) [inline], [explicit]`

Constructor.

Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 140 of file `for_each_selectors.h`.

4.117.3 Member Function Documentation

4.117.3.1 `template<typename _It, typename _Tp> bool __gnu_parallel::__replace_selector<_It, _Tp>::operator()(_Tp & __v, _It __i) [inline]`

Functor execution.

Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Definition at line 146 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_selector<_It, _Tp>::__new_val`.

4.117.4 Member Data Documentation

4.117.4.1 `template<typename _It, typename _Tp> const _Tp& __gnu_parallel::__replace_selector<_It, _Tp>::__new_val`

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp>::operator()()`.

4.117.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M.finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

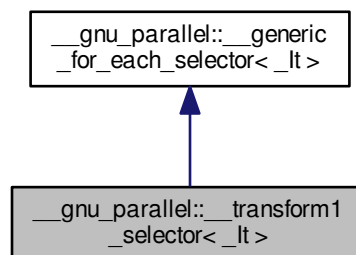
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.118 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.118.1 Detailed Description

`template<typename _It>struct __gnu_parallel::__transform1_selector<_It>`

`std::transform()` `__selector`, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

4.118.2 Member Function Documentation

4.118.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__transform1_selector<_It>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

4.118.3 Member Data Documentation

4.118.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

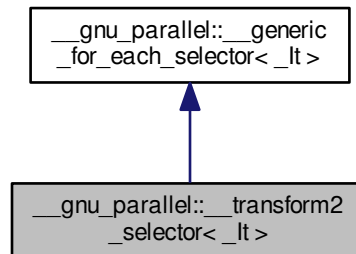
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.119 __gnu_parallel::__transform2_selector<_It> Struct Template Reference

Inheritance diagram for __gnu_parallel::__transform2_selector<_It>:



Public Member Functions

- `template<typename _Op >`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.119.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform2_selector<_It>`

`std::transform()` __selector, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

4.119.2 Member Function Documentation

4.119.2.1 `template<typename _It> template<typename _Op > bool __gnu_parallel::__transform2_selector<_It>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

4.119.3 Member Data Documentation

4.119.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector< _It >::M_finish_iterator`
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

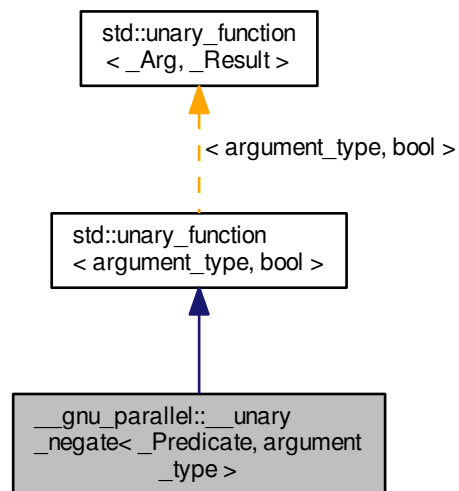
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.120 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__unary_negate< _Predicate, argument_type >`:



Public Types

- typedef [argument_type](#) `argument_type`
- typedef bool [result_type](#)

Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- bool `operator()` (`const argument_type &__x`)

Protected Attributes

- `_Predicate _M_pred`

4.120.1 Detailed Description

```
template<typename _Predicate, typename argument_type> class __gnu_parallel::__unary_negate<_Predicate, argument_type>
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file `parallel/base.h`.

4.120.2 Member Typedef Documentation

4.120.2.1 `typedef argument_type std::unary_function< argument_type, bool>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.120.2.2 `typedef bool std::unary_function< argument_type, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.121 `__gnu_parallel::__DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

Public Types

- typedef
 `_TraitsType::difference_type` `_DifferenceType`
- typedef `std::iterator_traits`
 `<_RAIter>` `_TraitsType`
- typedef `_TraitsType::value_type` `_ValueType`

Public Member Functions

- [_DRandomShufflingGlobalData](#) (`_RAIter &__source`)

Public Attributes

- `_ThreadIndex` * `_M_bin_proc`
- `_DifferenceType` ** `_M_dist`
- `int` `_M_num_bins`
- `int` `_M_num_bits`
- `_RAIter &` `_M_source`
- `_DifferenceType` * `_M_starts`
- `_ValueType` ** `_M_temporaries`

4.121.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>`

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 52 of file `random_shuffle.h`.

4.121.2 Constructor & Destructor Documentation

4.121.2.1 `template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__DRandomShufflingGlobalData(_RAIter & __source) [inline]`

Constructor.

Definition at line 83 of file `random_shuffle.h`.

4.121.3 Member Data Documentation

4.121.3.1 `template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_bin_proc`

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

4.121.3.2 `template<typename _RAIter> DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_dist`

Two-dimensional array to hold the thread-bin distribution.

Dimensions `(_M_num_threads + 1) __x (_M_num_bins + 1)`.

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

4.121.3.3 `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_num_bins`

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

4.121.3.4 `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_num_bits`

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

4.122 `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference 911

4.121.3.5 `template<typename _RAIter> _RAIter& __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

4.121.3.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

4.121.3.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

4.122 `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

Public Attributes

- [_BinIndex __bins_end](#)
- [_BinIndex __M_bins_begin](#)
- [int __M_num_threads](#)
- [_DRandomShufflingGlobalData<_RAIter> * __M_sd](#)
- [uint32_t __M_seed](#)

4.122.1 Detailed Description

`template<typename _RAIter, typename _RandomNumberGenerator> struct __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>`

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 91 of file `random_shuffle.h`.

4.122.2 Member Data Documentation

4.122.2.1 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::__bins_end`

End index for bins taken care of by this thread.

Definition at line 100 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

4.122.2.2 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::M_bins_begin`

Begin index for bins taken care of by this thread.

Definition at line 97 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

4.122.2.3 `template<typename _RAIter, typename _RandomNumberGenerator> int __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::M_num_threads`

Number of threads participating in total.

Definition at line 94 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.122.2.4 `template<typename _RAIter, typename _RandomNumberGenerator> _DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::M_sd`

Pointer to global data.

Definition at line 106 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.122.2.5 `template<typename _RAIter, typename _RandomNumberGenerator> uint32_t __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::M_seed`

Random _M_seed for this thread.

Definition at line 103 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

4.123 __gnu_parallel::_DummyReduct Struct Reference

Public Member Functions

- `bool operator() (bool, bool) const`

4.123.1 Detailed Description

Reduction function doing nothing.

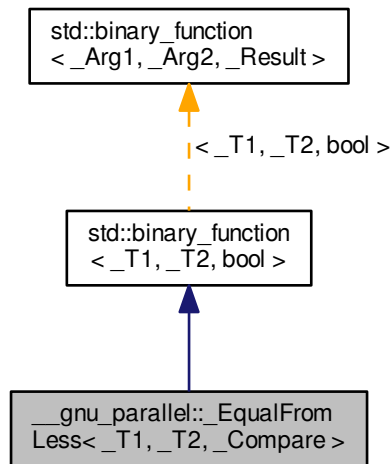
Definition at line 298 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.124 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- **`_EqualFromLess`** (`_Compare &__comp`)
- `bool operator()` (`const _T1 &__a, const _T2 &__b`)

4.124.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file `parallel/base.h`.

4.124.2 Member Typedef Documentation

4.124.2.1 `typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.124.2.2 `typedef bool std::binary_function<_T1, _T2, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.124.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

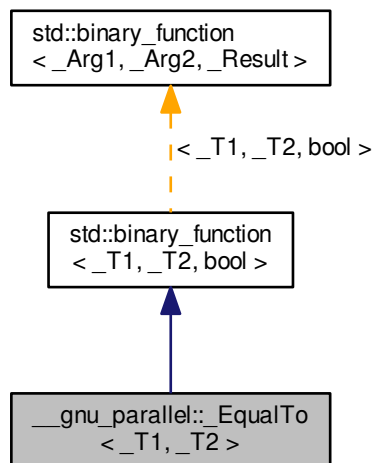
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.125 `__gnu_parallel::_EqualTo<_T1, _T2>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_EqualTo<_T1, _T2>`:



Public Types

- `typedef _T1` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _T2` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

4.125.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::EqualTo< _T1, _T2 >
```

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `parallel/base.h`.

4.125.2 Member Typedef Documentation

4.125.2.1 `typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.125.2.2 `typedef bool std::binary_function< _T1, _T2, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.125.2.3 `typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.126 `__gnu_parallel::_GuardedIterator<_RAIter, _Compare>` Class Template Reference

Public Member Functions

- `_GuardedIterator` (`_RAIter __begin`, `_RAIter __end`, `_Compare &__comp`)
- `operator _RAIter` ()
- `std::iterator_traits< _RAIter >::value_type & operator* ()`
- `_GuardedIterator< _RAIter, _Compare > & operator++ ()`

Friends

- `bool operator< (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`
- `bool operator<= (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`

4.126.1 Detailed Description

`template<typename _RAIter, typename _Compare> class __gnu_parallel::_GuardedIterator<_RAIter, _Compare>`

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 73 of file `multiway_merge.h`.

4.126.2 Constructor & Destructor Documentation

4.126.2.1 `template<typename _RAIter, typename _Compare> __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::_GuardedIterator(_RAIter __begin, _RAIter __end, _Compare & __comp) [inline]`

Constructor. Sets iterator to beginning of sequence.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator provided for associated overloaded compare operators.

Definition at line 91 of file `multiway_merge.h`.

4.126.3 Member Function Documentation

4.126.3.1 `template<typename _RAIter, typename _Compare> __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator _RAIter() [inline]`

Convert to wrapped iterator.

Returns

Wrapped iterator.

Definition at line 112 of file `multiway_merge.h`.

4.126.3.2 `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::value_type& __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator*() [inline]`

Dereference operator.

Returns

Referenced element.

Definition at line 107 of file `multiway_merge.h`.

4.126.3.3 `template<typename _RAIter, typename _Compare> _GuardedIterator<_RAIter, _Compare>& __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator++() [inline]`

Pre-increment operator.

Returns

This.

Definition at line 98 of file `multiway_merge.h`.

4.126.4 Friends And Related Function Documentation

4.126.4.1 `template<typename _RAIter, typename _Compare> bool operator< (_GuardedIterator< _RAIter, _Compare> & __bi1, _GuardedIterator< _RAIter, _Compare> & __bi2)` [`friend`]

Compare two elements referenced by guarded iterators.

Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

Returns

`true` if less.

Definition at line 120 of file `multiway_merge.h`.

4.126.4.2 `template<typename _RAIter, typename _Compare> bool operator<= (_GuardedIterator< _RAIter, _Compare> & __bi1, _GuardedIterator< _RAIter, _Compare> & __bi2)` [`friend`]

Compare two elements referenced by guarded iterators.

Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

Returns

`True` if less equal.

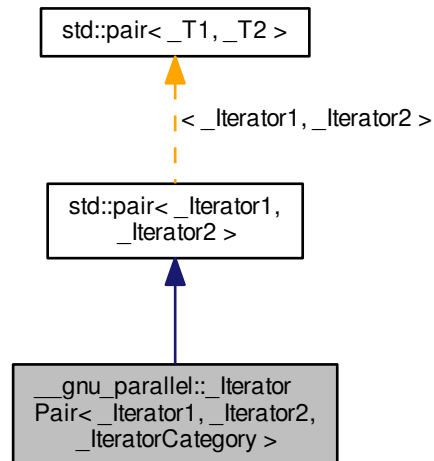
Definition at line 135 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

- [multiway_merge.h](#)

4.127 `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`:



Public Types

- `typedef std::iterator_traits< _Iterator1 > TraitsType`
- `typedef TraitsType::difference_type difference_type`
- `typedef _Iterator1 first_type`
- `typedef _IteratorCategory iterator_category`
- `typedef _IteratorPair * pointer`
- `typedef _IteratorPair & reference`
- `typedef _Iterator2 second_type`
- `typedef void value_type`

Public Member Functions

- `_IteratorPair` (`const _Iterator1 &__first, const _Iterator2 &__second`)
- `void __p first && noexcept` (`swap(second, __p.second)`)
- `operator _Iterator2` () `const`
- `_IteratorPair operator+` (`difference_type __delta`) `const`
- `_IteratorPair & operator++` ()
- `const _IteratorPair operator++` (`int`)
- `difference_type operator-` (`const _IteratorPair &__other`) `const`
- `_IteratorPair & operator--` ()
- `const _IteratorPair operator--` (`int`)
- `_IteratorPair & operator=` (`const _IteratorPair &__other`)
- `void swap` (`pair &__p`) `noexcept`(`noexcept(swap(first`

Public Attributes

- `_Iterator1` [first](#)
- [pair](#)
is_nothrow_move_assignable
<_Iterator2 >::value **first**
- `_Iterator2` [second](#)
- **second**
- return * **this**

4.127.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

4.127.2 Member Typedef Documentation

4.127.2.1 `typedef _Iterator2 std::pair<_Iterator1, _Iterator2 >::second_type` [\[inherited\]](#)

`first_type` is the first bound type

Definition at line 99 of file `stl_pair.h`.

4.127.3 Member Data Documentation

4.127.3.1 `_Iterator1 std::pair<_Iterator1, _Iterator2 >::first` [\[inherited\]](#)

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

4.127.3.2 `_Iterator2 std::pair<_Iterator1, _Iterator2 >::second` [\[inherited\]](#)

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

4.128 `__gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory >` Class Template Reference

Public Types

- `typedef std::iterator_traits<_Iterator1 >::difference_type` **difference_type**

- typedef `_IteratorCategory` **iterator_category**
- typedef `_IteratorTriple` * **pointer**
- typedef `_IteratorTriple` & **reference**
- typedef void **value_type**

Public Member Functions

- **_IteratorTriple** (const `_Iterator1` &__first, const `_Iterator2` &__second, const `_Iterator3` &__third)
- **operator _Iterator3** () const
- `_IteratorTriple` **operator+** (difference_type __delta) const
- `_IteratorTriple` & **operator++** ()
- const `_IteratorTriple` **operator++** (int)
- difference_type **operator-** (const `_IteratorTriple` &__other) const
- `_IteratorTriple` & **operator--** ()
- const `_IteratorTriple` **operator--** (int)
- `_IteratorTriple` & **operator=** (const `_IteratorTriple` &__other)

Public Attributes

- `_Iterator1` **_M_first**
- `_Iterator2` **_M_second**
- `_Iterator3` **_M_third**

4.128.1 Detailed Description

template<typename `_Iterator1`, typename `_Iterator2`, typename `_Iterator3`, typename `_IteratorCategory`>class `__gnu_parallel::IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >`

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

4.129 `__gnu_parallel::__Job< __DifferenceTp >` Struct Template Reference

Public Types

- typedef `_DifferenceTp` **_DifferenceType**

Public Attributes

- volatile `_DifferenceType` **_M_first**
- volatile `_DifferenceType` **_M_last**
- volatile `_DifferenceType` **_M_load**

4.129.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::__Job<_DifferenceTp>`

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

4.129.2 Member Data Documentation

4.129.2.1 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::_M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

4.129.2.2 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::_M_last`

Last element.

Changed by owning thread only.

Definition at line 67 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

4.129.2.3 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::_M_load`

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

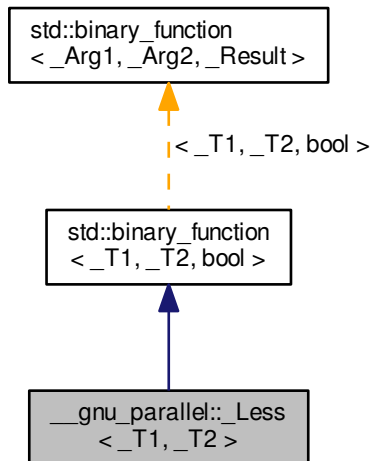
Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

4.130 `__gnu_parallel::_Less<_T1, _T2>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



Public Types

- typedef `_T1` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_T2` `second_argument_type`

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

4.130.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_Less<_T1, _T2>
```

Similar to `std::less`, but allows two different types.

Definition at line 252 of file `parallel/base.h`.

4.130.2 Member Typedef Documentation

4.130.2.1 typedef `_T1` `std::binary_function<_T1, _T2, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.130.2.2 `typedef bool std::binary_function<_T1, _T2, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.130.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

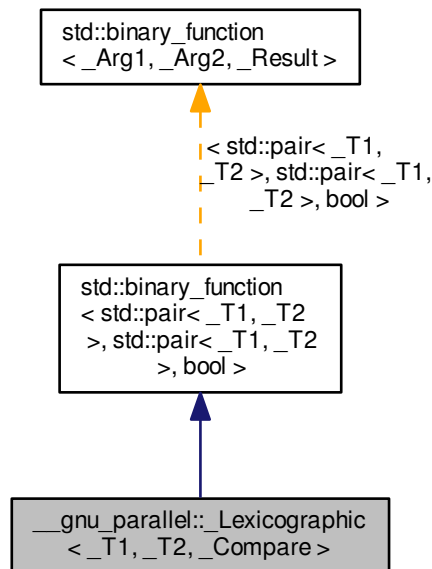
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.131 `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`:



Public Types

- `typedef std::pair<_T1, _T2> first_argument_type`
- `typedef bool result_type`
- `typedef std::pair<_T1, _T2> second_argument_type`

Public Member Functions

- `_Lexicographic` (`_Compare` & `__comp`)
- `bool operator()` (const `std::pair<_T1, _T2>` & `__p1`, const `std::pair<_T1, _T2>` & `__p2`) const

4.131.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::Lexicographic< _T1, _T2, _Compare >
```

Compare `__a` a pair of types lexicographically, ascending.

Definition at line 53 of file `multiseq_selection.h`.

4.131.2 Member Typedef Documentation

4.131.2.1 `typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.131.2.2 `typedef bool std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.131.2.3 `typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

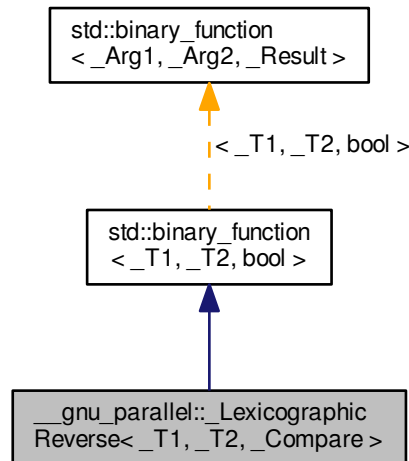
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [multiseq_selection.h](#)

4.132 `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`:



Public Types

- typedef `_T1` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_T2` `second_argument_type`

Public Member Functions

- **`_LexicographicReverse`** (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

4.132.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>
```

Compare __a pair of types lexicographically, descending.

Definition at line 80 of file `multiseq_selection.h`.

4.132.2 Member Typedef Documentation

4.132.2.1 typedef `_T1` `std::binary_function<_T1, _T2, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.132.2.2 `typedef bool std::binary_function< _T1, _T2, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.132.2.3 `typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

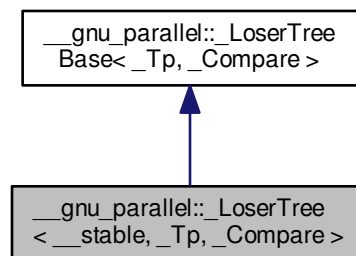
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [multiseq_selection.h](#)

4.133 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` & `__key`, int `__source`, bool `__sup`)

Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

4.133.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`

Stable `_LoserTree` variant.

Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 169 of file `losertree.h`.

4.133.2 Member Function Documentation

4.133.2.1 `template<bool __stable, typename _Tp, typename _Compare > void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 222 of file `losertree.h`.

4.133.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline],[inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

4.133.2.3 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (const _Tp & __key, int __source, bool __sup) [inline],[inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

4.133.3 Member Data Documentation

4.133.3.1 `template<typename _Tp, typename _Compare> _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_comp` [protected], [inherited]

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

4.133.3.2 `template<typename _Tp, typename _Compare> bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_first_insert` [protected], [inherited]

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.133.3.3 `template<typename _Tp, typename _Compare> unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_log_k` [protected], [inherited]

`log_2{M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.133.3.4 `template<typename _Tp, typename _Compare> _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_losers` [protected], [inherited]

`_LoserTree` elements.

Definition at line 75 of file `losertree.h`.

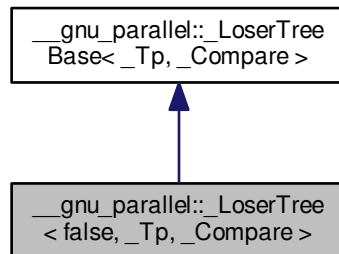
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.134 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)
- void [__delete_min_insert](#) (_Tp __key, bool __sup)
- int [__get_min_source](#) ()
- void [__init](#) ()
- unsigned int [__init_winner](#) (unsigned int __root)
- void [__insert_start](#) (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare [_M_comp](#)
- bool [_M_first_insert](#)
- unsigned int [_M_ik](#)
- unsigned int [_M_k](#)
- unsigned int [_M_log_k](#)
- [_Loser](#) * [_M_losers](#)
- unsigned int [_M_offset](#)

4.134.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTree< false, _Tp, _Compare >
```

Unstable `_LoserTree` variant.

Stability (non-stable here) is selected with partial specialization.

Definition at line 261 of file `losertree.h`.

4.134.2 Member Function Documentation

4.134.2.1 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::_delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

Parameters

<code>__key</code>	the <code>_M_key</code> to insert
<code>__sup</code>	true iff <code>__key</code> is an explicitly marked supremum

Definition at line 324 of file `losertree.h`.

4.134.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source () [inline],[inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

4.134.2.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner (unsigned int __root) [inline]`

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

Parameters

<code>__root</code>	<code>__index</code> of the "game" to start.
---------------------	--

Definition at line 284 of file `losertree.h`.

4.134.2.4 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start (const _Tp & __key, int __source, bool __sup) [inline],[inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

4.134.3 Member Data Documentation

4.134.3.1 `template<typename _Tp, typename _Compare> _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_comp` `[protected]`, `[inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

4.134.3.2 `template<typename _Tp, typename _Compare> bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_first_insert` `[protected]`, `[inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.134.3.3 `template<typename _Tp, typename _Compare> unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_log_k` `[protected]`, `[inherited]`

`log_2{M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.134.3.4 `template<typename _Tp, typename _Compare> _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::M_losers` `[protected]`, `[inherited]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

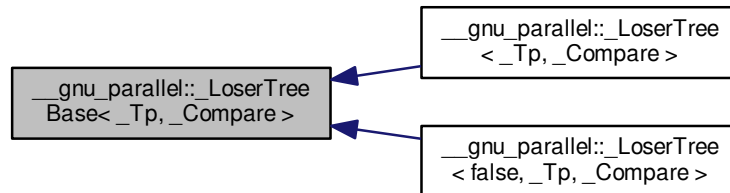
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.135 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



Classes

- struct `_Loser`

Internal representation of a `_LoserTree` element.

Public Member Functions

- `_LoserTreeBase` (unsigned int `__k`, `_Compare` `__comp`)
- `~_LoserTreeBase` ()
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

4.135.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

Parameters

<code>_Tp</code>	the element type
<code>_Compare</code>	the comparator to use, defaults to <code>std::less<_Tp></code>

Definition at line 55 of file `losertree.h`.

4.135.2 Constructor & Destructor Documentation

4.135.2.1 `template<typename _Tp, typename _Compare> __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__LoserTreeBase(unsigned int __k, _Compare __comp) [inline]`

The constructor.

Parameters

<code>__k</code>	The number of sequences to merge.
<code>__comp</code>	The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::__rd_log2()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_first_insert`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_log_k`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_losers`.

4.135.2.2 `template<typename _Tp, typename _Compare> __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::~~LoserTreeBase() [inline]`

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_losers`.

4.135.3 Member Function Documentation

4.135.3.1 `template<typename _Tp, typename _Compare> int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__get_min_source() [inline]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_losers`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__Loser::__M_source`.

4.135.3.2 `template<typename _Tp, typename _Compare> void __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__insert_start(const _Tp & __key, int __source, bool __sup) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

4.135.4 Member Data Documentation

4.135.4.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp` [protected]

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

4.135.4.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert` [protected]

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.135.4.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` [protected]

`log_2(_M_k)`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.135.4.4 `template<typename _Tp, typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` [protected]

`_LoserTree` __elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.136 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference

Public Attributes

- `_Tp` `_M_key`
- `int` `_M_source`
- `bool` `_M_sup`

4.136.1 Detailed Description

`template<typename _Tp, typename _Compare>struct __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser`

Internal representation of a `_LoserTree` element.

Definition at line 59 of file `losertree.h`.

4.136.2 Member Data Documentation

4.136.2.1 `template<typename _Tp, typename _Compare>_Tp __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_key`

`_M_key` of the element in the `_LoserTree`.

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`.

4.136.2.2 `template<typename _Tp, typename _Compare>int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_source`

`__index` of the `__source` `__sequence`.

Definition at line 64 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_get_min_source()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`.

4.136.2.3 `template<typename _Tp, typename _Compare>bool __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_sup`

flag, true iff this is a "maximum" `__sentinel`.

Definition at line 62 of file `losertree.h`.

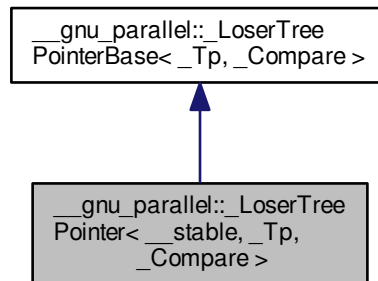
Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

4.137 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointer`** (unsigned int __k, _Compare __comp=[std::less](#)<_Tp>())
- void **`__delete_min_insert`** (const _Tp &__key, bool __sup)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int __root)
- void **`__insert_start`** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- [_Loser](#) * **`_M_losers`**
- unsigned int **`_M_offset`**

4.137.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `_LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.

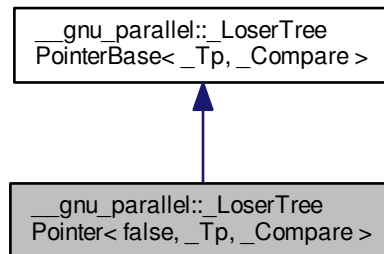
Definition at line 409 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.138 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointer` (unsigned int __k, _Compare __comp=[std::less](#)< _Tp >())
- void `__delete_min_insert` (const _Tp &__key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [_Loser](#) * `_M_losers`
- unsigned int `_M_offset`

4.138.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable `_LoserTree` implementation.

The stable variant is above.

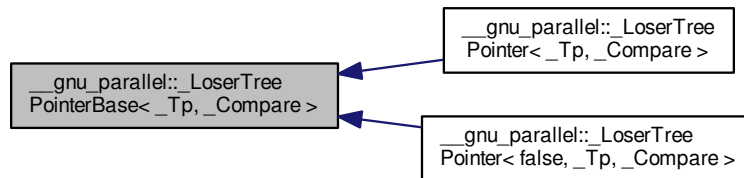
Definition at line 491 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.139 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>`:



Classes

- struct [_Loser](#)
Internal representation of `_LoserTree` __elements.

Public Member Functions

- `_LoserTreePointerBase` (unsigned int __k, _Compare __comp=[std::less](#)<_Tp>())
- int `__get_min_source` ()
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [_Loser](#) * `_M_losers`
- unsigned int `_M_offset`

4.139.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>
```

Base class of `_Loser` Tree implementation using pointers.

Definition at line 357 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.140 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser` Struct Reference

Public Attributes

- `const _Tp * _M_keyp`
- `int _M_source`
- `bool _M_sup`

4.140.1 Detailed Description

`template<typename _Tp, typename _Compare>struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`

Internal representation of `_LoserTree` __elements.

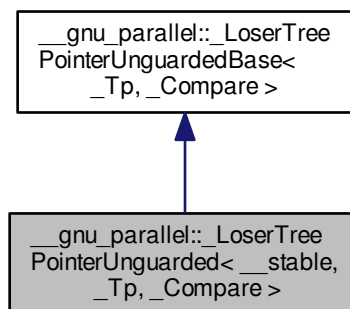
Definition at line 361 of file `losertree.h`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

4.141 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointerUnguarded` (`unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >()`)
- `void __delete_min_insert` (`const _Tp &__key, bool __sup`)
- `int __get_min_source` (`()`)
- `void __init` (`()`)
- `unsigned int __init_winner` (`unsigned int __root`)
- `void __insert_start` (`const _Tp &__key, int __source, bool`)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser` * **`_M_losers`**
- unsigned int **`_M_offset`**

4.141.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`

Stable unguarded `_LoserTree` variant storing pointers.

Unstable variant is implemented below using partial specialization.

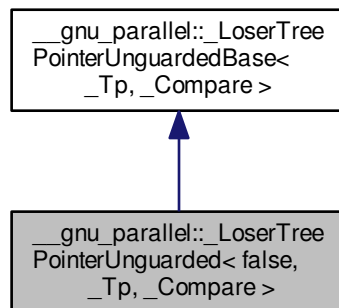
Definition at line 891 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.142 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp`=`std::less< _Tp >()`)
- void **`__delete_min_insert`** (const `_Tp` &`__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()

- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` & `__key`, int `__source`, bool)

Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

4.142.1 Detailed Description

`template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`

Unstable unguarded `_LoserTree` variant storing pointers.

Stable variant is above.

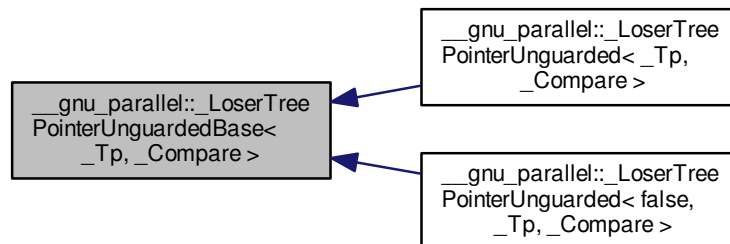
Definition at line 977 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.143 `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int `__k`, const `_Tp` & `__sentinel`, `_Compare` `__comp`=`std::less< _Tp >()`)
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` & `__key`, int `__source`, bool)

Protected Attributes

- `_Compare` **`_M_comp`**
- `unsigned int` **`_M_ik`**
- `unsigned int` **`_M_k`**
- `_Loser *` **`_M_losers`**
- `unsigned int` **`_M_offset`**

4.143.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 828 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.144 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference

Static Public Attributes

- `static const bool` [_M_use_pointer](#)

4.144.1 Detailed Description

```
template<typename _Tp>struct __gnu_parallel::_LoserTreeTraits< _Tp >
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
```

```
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

Parameters

<code>_Tp</code>	type to give the loser tree traits for.
------------------	---

Definition at line 731 of file multiway_merge.h.

4.144.2 Member Data Documentation

4.144.2.1 `template<typename _Tp> const bool __gnu_parallel::_LoserTreeTraits< _Tp >::M_use_pointer` [static]

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

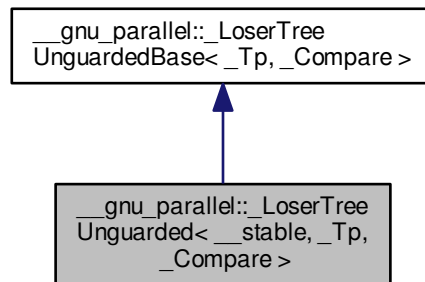
Definition at line 739 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.145 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreeUnguarded`** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void **`__delete_min_insert`** (_Tp __key, bool)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int __root)
- void **`__insert_start`** (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser` * **`_M_losers`**
- unsigned int **`_M_offset`**

4.145.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded `_LoserTree`.

Unstable variant is selected below with partial specialization.

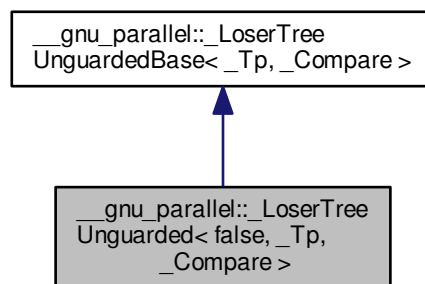
Definition at line 646 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.146 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreeUnguarded`** (unsigned int `__k`, const `_Tp` & `__sentinel`, `_Compare` `__comp`=`std::less< _Tp >()`)
- void **`__delete_min_insert`** (`_Tp` `__key`, bool)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` & `__key`, int `__source`, bool)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser` * **`_M_losers`**
- unsigned int **`_M_offset`**

4.146.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::__LoserTreeUnguarded< false, _Tp, _Compare >
```

Non-Stable implementation of unguarded `_LoserTree`.

Stable implementation is above.

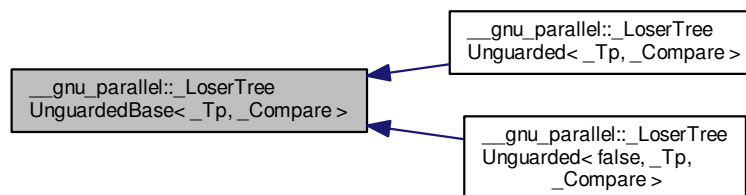
Definition at line 734 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.147 `__gnu_parallel::__LoserTreeUnguardedBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__LoserTreeUnguardedBase< _Tp, _Compare >`:



Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int `__k`, const `_Tp` & `__sentinel`, `_Compare` `__comp`=`std::less< _Tp >()`)
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` & `__key`, int `__source`, bool)

Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

4.147.1 Detailed Description

```
template<typename _Tp, typename _Compare>class __gnu_parallel::__LoserTreeUnguardedBase< _Tp, _Compare >
```

Base class for unguarded `_LoserTree` implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

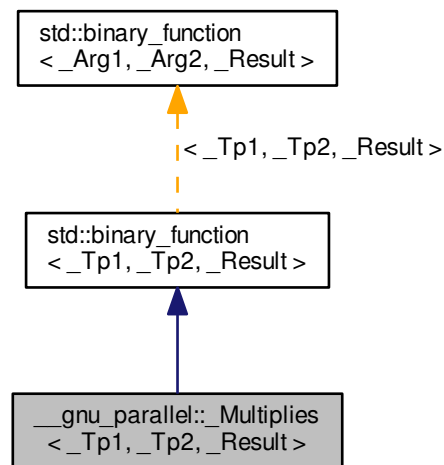
Definition at line 574 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.148 `__gnu_parallel::__Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__Multiplies<_Tp1, _Tp2, _Result>`:



Public Types

- `typedef _Tp1` [first_argument_type](#)
- `typedef _Result` [result_type](#)
- `typedef _Tp2` [second_argument_type](#)

Public Member Functions

- `_Result` **operator()** (const `_Tp1` &`_x`, const `_Tp2` &`_y`) const

4.148.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>()) * (*static_cast<_Tp2*>())>struct
__gnu_parallel::Multiplies< _Tp1, _Tp2, _Result >
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `parallel/base.h`.

4.148.2 Member Typedef Documentation

4.148.2.1 `typedef _Tp1 std::binary_function< _Tp1, _Tp2, _Result >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.148.2.2 `typedef _Result std::binary_function< _Tp1, _Tp2, _Result >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.148.2.3 `typedef _Tp2 std::binary_function< _Tp1, _Tp2, _Result >::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.149 `__gnu_parallel::Nothing` Struct Reference

Public Member Functions

- `template<typename _It >`
void [operator\(\)](#) (`_It __i`)

4.149.1 Detailed Description

Functor doing nothing.

For some `__reduction` tasks (this is not a function object, but is passed as `__selector __dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

4.149.2 Member Function Documentation

4.149.2.1 `template<typename _It > void __gnu_parallel::Nothing::operator() (_It __i)` `[inline]`

Functor execution.

Parameters

<code>__i</code>	iterator referencing object.
------------------	------------------------------

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.150 `__gnu_parallel::_Piece<_DifferenceTp>` Struct Template Reference

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Attributes

- `_DifferenceType _M_begin`
- `_DifferenceType _M_end`

4.150.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Piece<_DifferenceTp>`

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

4.150.2 Member Data Documentation

4.150.2.1 `template<typename _DifferenceTp> _DifferenceType __gnu_parallel::_Piece<_DifferenceTp>::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

4.150.2.2 `template<typename _DifferenceTp> _DifferenceType __gnu_parallel::_Piece<_DifferenceTp>::_M_end`

End of subsequence.

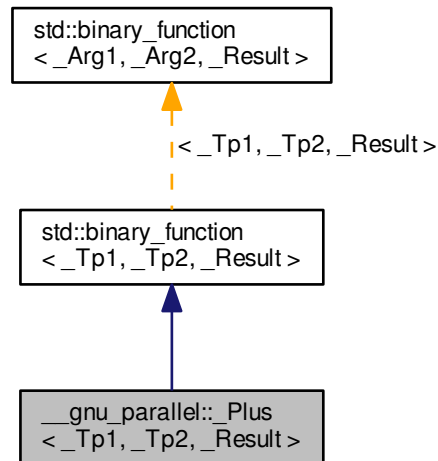
Definition at line 54 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.151 `__gnu_parallel::Plus<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::Plus<_Tp1, _Tp2, _Result>`:



Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Result` `result_type`
- typedef `_Tp2` `second_argument_type`

Public Member Functions

- `_Result` **operator()** (const `_Tp1` &`_x`, const `_Tp2` &`_y`) const

4.151.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0)))> struct
__gnu_parallel::Plus<_Tp1, _Tp2, _Result>
```

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `parallel/base.h`.

4.151.2 Member Typedef Documentation

4.151.2.1 typedef `_Tp1` `std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.151.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.151.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.152 `__gnu_parallel::__PMWSSortingData<_RAIter>` Struct Template Reference

Public Types

- `typedef`
 `_TraitsType::difference_type` **`_DifferenceType`**
- `typedef std::iterator_traits`
 `<_RAIter>` **`_TraitsType`**
- `typedef _TraitsType::value_type` **`_ValueType`**

Public Attributes

- [_ThreadIndex](#) [_M_num_threads](#)
- `_DifferenceType` * [_M_offsets](#)
- `std::vector<_Piece`
 `<_DifferenceType>` * [_M_pieces](#)
- `_ValueType` * [_M_samples](#)
- `_RAIter` [_M_source](#)
- `_DifferenceType` * [_M_starts](#)
- `_ValueType` ** [_M_temporary](#)

4.152.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::__PMWSSortingData<_RAIter>`

Data accessed by all threads.

PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

4.152.2 Member Data Documentation

4.152.2.1 `template<typename _RAIter> _ThreadIndex __gnu_parallel::__PMWSSortingData<_RAIter>::_M_num_threads`

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.152.2.2 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWMSSortingData<_RAIter>::M_offsets`

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

4.152.2.3 `template<typename _RAIter> std::vector<_Piece<_DifferenceType>>* __gnu_parallel::PMWMSSortingData<_RAIter>::M_pieces`

Pieces of data to merge [thread][__sequence].

Definition at line 86 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.152.2.4 `template<typename _RAIter> _ValueType* __gnu_parallel::PMWMSSortingData<_RAIter>::M_samples`

Samples.

Definition at line 80 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, and `__gnu_parallel::parallel_sort_mwms()`.

4.152.2.5 `template<typename _RAIter> _RAIter __gnu_parallel::PMWMSSortingData<_RAIter>::M_source`

Input __begin.

Definition at line 71 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.152.2.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWMSSortingData<_RAIter>::M_starts`

Start indices, per thread.

Definition at line 74 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.152.2.7 `template<typename _RAIter> _ValueType** __gnu_parallel::PMWMSSortingData<_RAIter>::M_temporary`

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.153 `__gnu_parallel::PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**
- typedef `_PseudoSequenceIterator<_Tp, uint64_t>` **`iterator`**

Public Member Functions

- `_PseudoSequence` (`const _Tp &__val, _DifferenceType __count`)
- `iterator begin` () const
- `iterator end` () const

4.153.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>`

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

4.153.2 Constructor & Destructor Documentation

4.153.2.1 `template<typename _Tp, typename _DifferenceTp> __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::_PseudoSequence (const _Tp &__val, _DifferenceType __count) [inline]`

Constructor.

Parameters

<code>__val</code>	Element of the sequence.
<code>__count</code>	Number of (virtual) copies.

Definition at line 371 of file `parallel/base.h`.

4.153.3 Member Function Documentation

4.153.3.1 `template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::begin () const [inline]`

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

4.153.3.2 `template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::end () const [inline]`

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.154 `__gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>` Class Template Reference

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Member Functions

- `_PseudoSequenceliterator` (const `_Tp` &__val, `_DifferenceType` __pos)
- `bool operator!=` (const `_PseudoSequenceliterator` &__i2)
- `const _Tp &operator*` () const
- `_PseudoSequenceliterator &operator++` ()
- `_PseudoSequenceliterator operator++` (int)
- `_DifferenceType operator-` (const `_PseudoSequenceliterator` &__i2)
- `bool operator==` (const `_PseudoSequenceliterator` &__i2)
- `const _Tp &operator[]` (`_DifferenceType`) const

4.154.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

Parameters

<code>_Tp</code>	Sequence <code>_M</code> value type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.155 `__gnu_parallel::__QSBThreadLocal<_RAIter>` Struct Template Reference

Public Types

- `typedef`
`_TraitsType::difference_type _DifferenceType`
- `typedef` `std::pair<_RAIter,`
`_RAIter> _Piece`

- `typedef std::iterator_traits<_RAIter> _TraitsType`

Public Member Functions

- [`_QSBThreadLocal`](#) (int `__queue_size`)

Public Attributes

- `volatile _DifferenceType * _M_elements_leftover`
- [`_Piece _M_global`](#)
- [`_Piece _M_initial`](#)
- [`_RestrictedBoundedConcurrentQueue<_Piece> _M_leftover_parts`](#)
- [`_ThreadIndex _M_num_threads`](#)

4.155.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_QSBThreadLocal<_RAIter>`

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

4.155.2 Member Typedef Documentation

4.155.2.1 `template<typename _RAIter> typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_Piece`

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

4.155.3 Constructor & Destructor Documentation

4.155.3.1 `template<typename _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_QSBThreadLocal (int __queue_size) [inline]`

Constructor.

Parameters

<code>__queue_size</code>	size of the work-stealing queue.
---------------------------	----------------------------------

Definition at line 88 of file `balanced_quicksort.h`.

4.155.4 Member Data Documentation

4.155.4.1 `template<typename _RAIter> volatile _DifferenceType* __gnu_parallel::QSBThreadLocal<_RAIter>::M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.155.4.2 `template<typename _RAIter> _Piece __gnu_parallel::QSBThreadLocal<_RAIter>::M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

4.155.4.3 `template<typename _RAIter> _Piece __gnu_parallel::QSBThreadLocal<_RAIter>::M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.155.4.4 `template<typename _RAIter> _RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::QSBThreadLocal<_RAIter>::M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.155.4.5 `template<typename _RAIter> _ThreadIndex __gnu_parallel::QSBThreadLocal<_RAIter>::M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced_quicksort.h](#)

4.156 `__gnu_parallel::RandomNumber` Class Reference

Public Member Functions

- [_RandomNumber](#) ()
- [_RandomNumber](#) (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long [__genrand_bits](#) (int __bits)
- uint32_t [operator\(\)](#) ()
- uint32_t [operator\(\)](#) (uint64_t local_supremum)

4.156.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

4.156.2 Constructor & Destructor Documentation

4.156.2.1 __gnu_parallel::_RandomNumber::RandomNumber () [inline]

Default constructor. Seed with 0.

Definition at line 74 of file random_number.h.

4.156.2.2 __gnu_parallel::_RandomNumber::RandomNumber (uint32_t __seed, uint64_t *M_supremum* = 0x100000000ULL) [inline]

Constructor.

Parameters

<i>__seed</i>	Random <i>__seed</i> .
<i>_M_supremum</i>	Generate integer random numbers in the interval [0, <i>M_supremum</i>).

Definition at line 85 of file random_number.h.

4.156.3 Member Function Documentation

4.156.3.1 unsigned long __gnu_parallel::_RandomNumber::_genrand_bits (int *__bits*) [inline]

Generate a number of random bits, run-time parameter.

Parameters

<i>__bits</i>	Number of bits to generate.
---------------	-----------------------------

Definition at line 109 of file random_number.h.

4.156.3.2 uint32_t __gnu_parallel::_RandomNumber::operator() () [inline]

Generate unsigned random 32-bit integer.

Definition at line 94 of file random_number.h.

4.156.3.3 uint32_t __gnu_parallel::_RandomNumber::operator() (uint64_t *local_supremum*) [inline]

Generate unsigned random 32-bit integer in the interval [0, *local_supremum*).

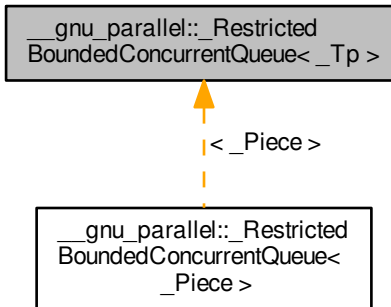
Definition at line 100 of file random_number.h.

The documentation for this class was generated from the following file:

- [random_number.h](#)

4.157 `__gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>`:



Public Member Functions

- `_RestrictedBoundedConcurrentQueue` (`_SequenceIndex __max_size`)
- `~_RestrictedBoundedConcurrentQueue` ()
- `bool pop_back` (`_Tp &__t`)
- `bool pop_front` (`_Tp &__t`)
- `void push_front` (`const _Tp &__t`)

4.157.1 Detailed Description

`template<typename _Tp> class __gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>`

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

<code>_Tp</code>	Contained element type.
------------------	-------------------------

Definition at line 52 of file `queue.h`.

4.157.2 Constructor & Destructor Documentation

4.157.2.1 `template<typename _Tp> __gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>::__RestrictedBoundedConcurrentQueue (_SequenceIndex __max_size) [inline]`

Constructor. Not to be called concurrent, of course.

Parameters

<code>__max_size</code>	Maximal number of elements to be contained.
-------------------------	---

Definition at line 68 of file `queue.h`.

4.157.2.2 `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::~~_RestrictedBoundedConcurrentQueue () [inline]`

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file `queue.h`.

4.157.3 Member Function Documentation

4.157.3.1 `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (_Tp & __t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 127 of file `queue.h`.

4.157.3.2 `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (_Tp & __t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 100 of file `queue.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.157.3.3 `template<typename _Tp> void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (const _Tp & __t) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 83 of file `queue.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this class was generated from the following file:

- [queue.h](#)

4.158 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Public Member Functions

- void **operator()** (`_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp`)

4.158.1 Detailed Description

`template<bool __stable, class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1007 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.159 `__gnu_parallel::SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Public Member Functions

- void **operator()** (`_RAIter __first`, `_RAIter __last`, `_StrictWeakOrdering __comp`)

4.159.1 Detailed Description

`template<class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-`__stable` sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1020 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.160 `__gnu_parallel::Settings` Struct Reference

Static Public Member Functions

- static const `_Settings` & `get` () throw ()
- static void `set` (`_Settings` &) throw ()

Public Attributes

- `_SequenceIndex` `accumulate_minimal_n`
- unsigned int `adjacent_difference_minimal_n`
- `_AlgorithmStrategy` `algorithm_strategy`
- unsigned int `cache_line_size`
- `_SequenceIndex` `count_minimal_n`
- `_SequenceIndex` `fill_minimal_n`
- `_FindAlgorithm` `find_algorithm`
- double `find_increasing_factor`
- `_SequenceIndex` `find_initial_block_size`
- `_SequenceIndex` `find_maximum_block_size`
- float `find_scale_factor`
- `_SequenceIndex` `find_sequential_search_size`
- `_SequenceIndex` `for_each_minimal_n`
- `_SequenceIndex` `generate_minimal_n`
- unsigned long long `L1_cache_size`

- unsigned long long `L2_cache_size`
- `_SequenceIndex` `max_element_minimal_n`
- `_SequenceIndex` `merge_minimal_n`
- unsigned int `merge_oversampling`
- `_SplittingAlgorithm` **`merge_splitting`**
- `_SequenceIndex` `min_element_minimal_n`
- `_MultiwayMergeAlgorithm` **`multiway_merge_algorithm`**
- int `multiway_merge_minimal_k`
- `_SequenceIndex` `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- `_SplittingAlgorithm` **`multiway_merge_splitting`**
- `_SequenceIndex` `nth_element_minimal_n`
- `_SequenceIndex` `partial_sort_minimal_n`
- `_PartialSumAlgorithm` **`partial_sum_algorithm`**
- float `partial_sum_dilation`
- unsigned int `partial_sum_minimal_n`
- double `partition_chunk_share`
- `_SequenceIndex` `partition_chunk_size`
- `_SequenceIndex` `partition_minimal_n`
- `_SequenceIndex` `qsb_steals`
- unsigned int `random_shuffle_minimal_n`
- `_SequenceIndex` `replace_minimal_n`
- `_SequenceIndex` `search_minimal_n`
- `_SequenceIndex` `set_difference_minimal_n`
- `_SequenceIndex` `set_intersection_minimal_n`
- `_SequenceIndex` `set_symmetric_difference_minimal_n`
- `_SequenceIndex` `set_union_minimal_n`
- `_SortAlgorithm` **`sort_algorithm`**
- `_SequenceIndex` `sort_minimal_n`
- unsigned int `sort_mwms_oversampling`
- unsigned int `sort_qs_num_samples_preset`
- `_SequenceIndex` `sort_qsb_base_case_maximal_n`
- `_SplittingAlgorithm` **`sort_splitting`**
- unsigned int `TLB_size`
- `_SequenceIndex` `transform_minimal_n`
- `_SequenceIndex` `unique_copy_minimal_n`
- `_SequenceIndex` **`workstealing_chunk_size`**

4.160.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file `settings.h`.

4.160.2 Member Function Documentation

4.160.2.1 `static const Settings& __gnu_parallel::Settings::get () throw () [static]`

Get the global settings.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partial_sum_linear()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs_conquer()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.160.2.2 `static void __gnu_parallel::Settings::set (Settings &) throw () [static]`

Set the global settings.

4.160.3 Member Data Documentation

4.160.3.1 `_SequenceIndex __gnu_parallel::Settings::accumulate_minimal_n`

Minimal input size for accumulate.

Definition at line 139 of file settings.h.

4.160.3.2 `unsigned int __gnu_parallel::Settings::adjacent_difference_minimal_n`

Minimal input size for adjacent_difference.

Definition at line 142 of file settings.h.

4.160.3.3 `unsigned int __gnu_parallel::Settings::cache_line_size`

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Definition at line 265 of file settings.h.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

4.160.3.4 `_SequenceIndex __gnu_parallel::Settings::count_minimal_n`

Minimal input size for count and count_if.

Definition at line 145 of file settings.h.

4.160.3.5 `_SequenceIndex __gnu_parallel::Settings::fill_minimal_n`

Minimal input size for fill.

Definition at line 148 of file settings.h.

4.160.3.6 `double __gnu_parallel::Settings::find_increasing_factor`

Block size increase factor for find.

Definition at line 151 of file settings.h.

4.160.3.7 `__SequenceIndex __gnu_parallel::Settings::find_initial_block_size`

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

4.160.3.8 `__SequenceIndex __gnu_parallel::Settings::find_maximum_block_size`

Maximal block size for find.

Definition at line 157 of file settings.h.

4.160.3.9 `float __gnu_parallel::Settings::find_scale_factor`

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

4.160.3.10 `__SequenceIndex __gnu_parallel::Settings::find_sequential_search_size`

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

4.160.3.11 `__SequenceIndex __gnu_parallel::Settings::for_each_minimal_n`

Minimal input size for `for_each`.

Definition at line 163 of file settings.h.

4.160.3.12 `__SequenceIndex __gnu_parallel::Settings::generate_minimal_n`

Minimal input size for `generate`.

Definition at line 166 of file settings.h.

4.160.3.13 `unsigned long long __gnu_parallel::Settings::L1_cache_size`

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

4.160.3.14 `unsigned long long __gnu_parallel::Settings::L2_cache_size`

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

4.160.3.15 `__SequenceIndex __gnu_parallel::Settings::max_element_minimal_n`

Minimal input size for `max_element`.

Definition at line 169 of file settings.h.

4.160.3.16 __SequenceIndex __gnu_parallel::_Settings::merge_minimal_n

Minimal input size for merge.

Definition at line 172 of file settings.h.

4.160.3.17 unsigned int __gnu_parallel::_Settings::merge_oversampling

Oversampling factor for merge.

Definition at line 175 of file settings.h.

Referenced by __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiway_merge().

4.160.3.18 __SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n

Minimal input size for min_element.

Definition at line 178 of file settings.h.

4.160.3.19 int __gnu_parallel::_Settings::multiway_merge_minimal_k

Oversampling factor for multiway_merge.

Definition at line 184 of file settings.h.

4.160.3.20 __SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n

Minimal input size for multiway_merge.

Definition at line 181 of file settings.h.

4.160.3.21 unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling

Oversampling factor for multiway_merge.

Definition at line 187 of file settings.h.

4.160.3.22 __SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n

Minimal input size for nth_element.

Definition at line 190 of file settings.h.

Referenced by __gnu_parallel::__parallel_nth_element().

4.160.3.23 __SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n

Minimal input size for partial_sort.

Definition at line 203 of file settings.h.

4.160.3.24 float __gnu_parallel::_Settings::partial_sum_dilation

Ratio for partial_sum. Assume "sum and write result" to be this factor slower than just "sum".

Definition at line 207 of file settings.h.

Referenced by __gnu_parallel::__parallel_partial_sum_linear().

4.160.3.25 unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n

Minimal input size for partial_sum.

Definition at line 210 of file settings.h.

4.160.3.26 `double __gnu_parallel::Settings::partition_chunk_share`

Chunk size for partition, relative to input size. If > 0.0 , this value overrides `partition_chunk_size`.

Definition at line 197 of file settings.h.

Referenced by `__gnu_parallel::__parallel_partition()`.

4.160.3.27 `__SequenceIndex __gnu_parallel::Settings::partition_chunk_size`

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by `__gnu_parallel::__parallel_partition()`.

4.160.3.28 `__SequenceIndex __gnu_parallel::Settings::partition_minimal_n`

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by `__gnu_parallel::__parallel_nth_element()`.

4.160.3.29 `__SequenceIndex __gnu_parallel::Settings::qsb_steals`

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

4.160.3.30 `unsigned int __gnu_parallel::Settings::random_shuffle_minimal_n`

Minimal input size for `random_shuffle`.

Definition at line 213 of file settings.h.

4.160.3.31 `__SequenceIndex __gnu_parallel::Settings::replace_minimal_n`

Minimal input size for `replace` and `replace_if`.

Definition at line 216 of file settings.h.

4.160.3.32 `__SequenceIndex __gnu_parallel::Settings::search_minimal_n`

Minimal input size for `search` and `search_n`.

Definition at line 273 of file settings.h.

4.160.3.33 `__SequenceIndex __gnu_parallel::Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Definition at line 219 of file settings.h.

4.160.3.34 `__SequenceIndex __gnu_parallel::Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file settings.h.

4.160.3.35 `_SequenceIndex __gnu_parallel::Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

4.160.3.36 `_SequenceIndex __gnu_parallel::Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

4.160.3.37 `_SequenceIndex __gnu_parallel::Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

4.160.3.38 `unsigned int __gnu_parallel::Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.160.3.39 `unsigned int __gnu_parallel::Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

4.160.3.40 `_SequenceIndex __gnu_parallel::Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. Applies to `std::sort` with dynamically load-balanced quicksort.

Definition at line 241 of file `settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.160.3.41 `unsigned int __gnu_parallel::Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

4.160.3.42 `_SequenceIndex __gnu_parallel::Settings::transform_minimal_n`

Minimal input size for parallel `std::transform`.

Definition at line 244 of file `settings.h`.

4.160.3.43 `_SequenceIndex __gnu_parallel::Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Definition at line 247 of file `settings.h`.

The documentation for this struct was generated from the following file:

- [settings.h](#)

4.161 `__gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

4.161.1 Detailed Description

`template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator> struct __gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >`

Split consistently.

Definition at line 122 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.162 `__gnu_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

Public Member Functions

- `void operator() (const _ThreadIndex __iam, _PMWMSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const`

4.162.1 Detailed Description

`template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator> struct __gnu_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`

Split by sampling.

Definition at line 187 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.163 `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

Public Member Functions

- `void operator() (const _ThreadIndex __iam, _PMWMSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const`

4.163.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< true,
_RAlter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

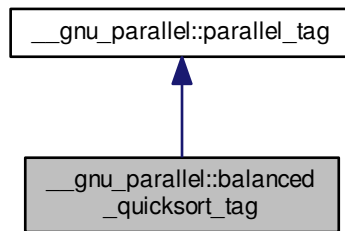
Definition at line 128 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.164 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



Public Member Functions

- **`balanced_quicksort_tag`** ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

4.164.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file `tags.h`.

4.164.2 Member Function Documentation

4.164.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.164.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline],[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

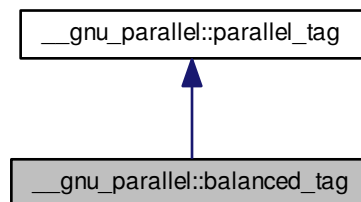
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.165 `__gnu_parallel::balanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_tag`:

**Public Member Functions**

- [_ThreadIndex](#) `__get_num_threads ()`
- void `set_num_threads (_ThreadIndex __num_threads)`

4.165.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file `tags.h`.

4.165.2 Member Function Documentation

4.165.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::get_num_threads ()` `[inline],[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.165.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline],[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

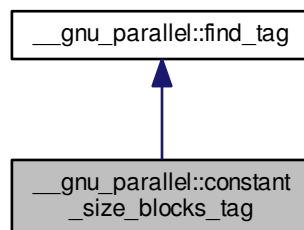
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.166 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



4.166.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See Also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

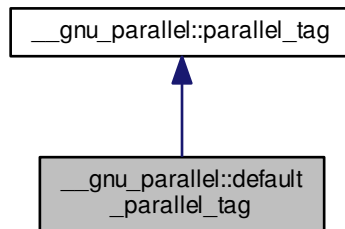
Definition at line 178 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.167 `__gnu_parallel::default_parallel_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



Public Member Functions

- **`default_parallel_tag`** (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` __num_threads)

4.167.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file `tags.h`.

4.167.2 Member Function Documentation

4.167.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.167.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

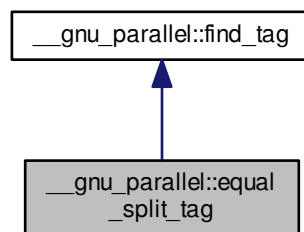
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.168 `__gnu_parallel::equal_split_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::equal_split_tag`:



4.168.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

See Also

`_GLIBCXX_FIND_EQUAL_SPLIT`

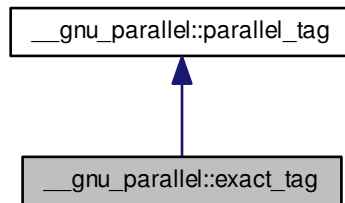
Definition at line 182 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.169 `__gnu_parallel::exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::exact_tag`:



Public Member Functions

- **`exact_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.169.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file `tags.h`.

4.169.2 Member Function Documentation

4.169.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.169.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex` `__num_threads`) `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

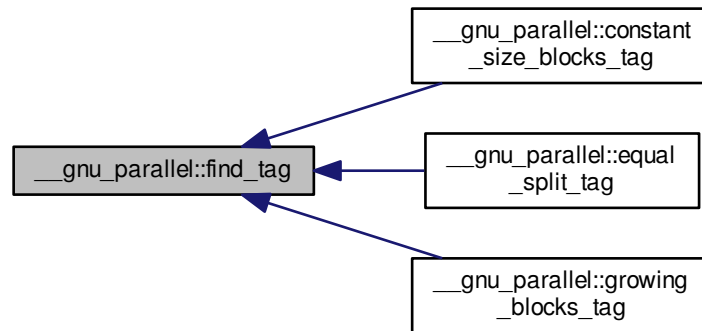
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.170 `__gnu_parallel::find_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::find_tag`:



4.170.1 Detailed Description

Base class for for `std::find()` variants.

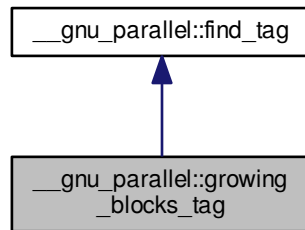
Definition at line 104 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.171 `__gnu_parallel::growing_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



4.171.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See Also

`_GLIBCXX_FIND_GROWING_BLOCKS`

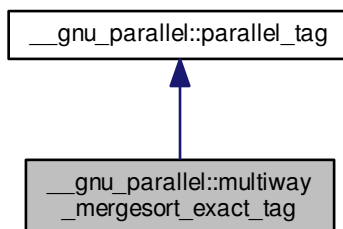
Definition at line 174 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.172 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



Public Member Functions

- `multiway_mergesort_exact_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

4.172.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file `tags.h`.

4.172.2 Member Function Documentation

4.172.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` () `[inline],[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.172.2.2 `void __gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex __num_threads`) `[inline],[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

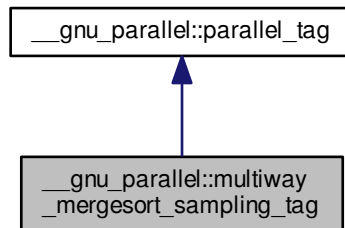
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.173 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



Public Member Functions

- **`multiway_mergesort_sampling_tag`** ([_ThreadIndex](#) `__num_threads`)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) `__num_threads`)

4.173.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file `tags.h`.

4.173.2 Member Function Documentation

4.173.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () [\[inline\]](#), [\[inherited\]](#)

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.173.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` ([_ThreadIndex](#) `__num_threads`) [\[inline\]](#), [\[inherited\]](#)

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.
-------------------------------	----------------------------

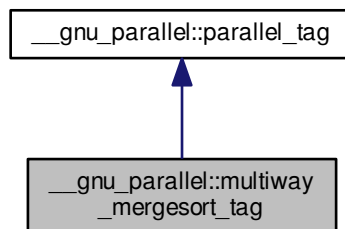
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.174 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



Public Member Functions

- **`multiway_mergesort_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.174.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file `tags.h`.

4.174.2 Member Function Documentation

4.174.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.174.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline],[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

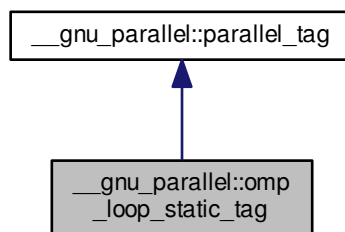
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.175 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



Public Member Functions

- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

4.175.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file `tags.h`.

4.175.2 Member Function Documentation

4.175.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` `[inline],[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.175.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline],[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

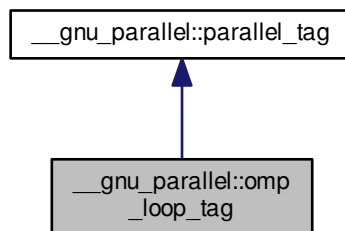
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.176 `__gnu_parallel::omp_loop_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:

**Public Member Functions**

- [_ThreadIndex __get_num_threads \(\)](#)
- `void set_num_threads (_ThreadIndex __num_threads)`

4.176.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

4.176.2 Member Function Documentation

4.176.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::get_num_threads ()` `[inline],[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.176.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline],[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

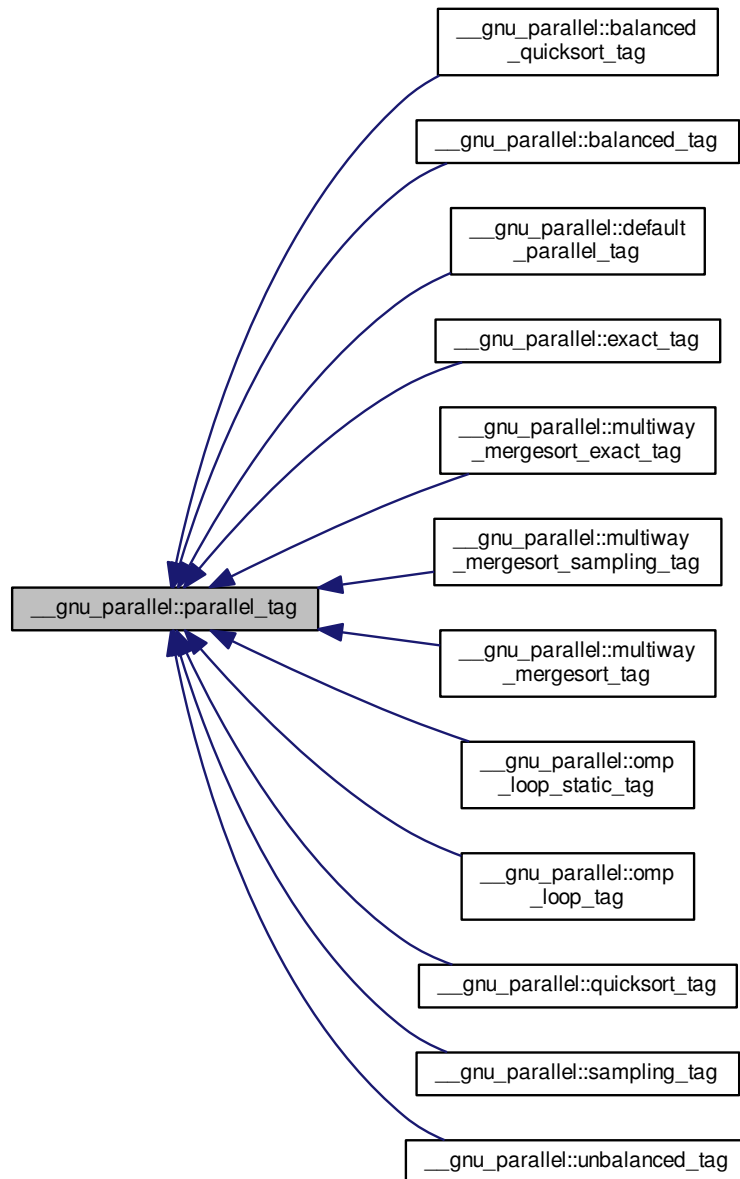
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.177 __gnu_parallel::parallel_tag Struct Reference

Inheritance diagram for __gnu_parallel::parallel_tag:



Public Member Functions

- [parallel_tag \(\)](#)
- [parallel_tag \(_ThreadIndex __num_threads\)](#)
- [_ThreadIndex __get_num_threads \(\)](#)

- void [set_num_threads](#) ([_ThreadIndex](#) __num_threads)

4.177.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

4.177.2 Constructor & Destructor Documentation

4.177.2.1 `__gnu_parallel::parallel_tag::parallel_tag ()` `[inline]`

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

4.177.2.2 `__gnu_parallel::parallel_tag::parallel_tag (_ThreadIndex __num_threads)` `[inline]`

Default constructor. Recommend number of threads to use.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

Definition at line 58 of file tags.h.

4.177.3 Member Function Documentation

4.177.3.1 `_ThreadIndex __gnu_parallel::parallel_tag::get_num_threads ()` `[inline]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.177.3.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

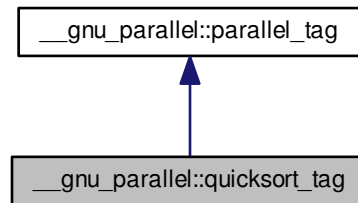
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.178 `__gnu_parallel::quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::quicksort_tag`:



Public Member Functions

- `quicksort_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.178.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file `tags.h`.

4.178.2 Member Function Documentation

4.178.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

4.178.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex` `__num_threads`) `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

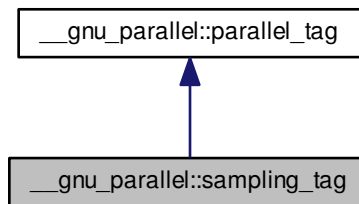
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.179 `__gnu_parallel::sampling_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::sampling_tag`:



Public Member Functions

- **sampling_tag** ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) **__get_num_threads** ()
- void **set_num_threads** ([_ThreadIndex](#) __num_threads)

4.179.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

4.179.2 Member Function Documentation

4.179.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads ()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.179.2.2 void `__gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.180 `__gnu_parallel::sequential_tag` Struct Reference

4.180.1 Detailed Description

Forces sequential execution at compile time.

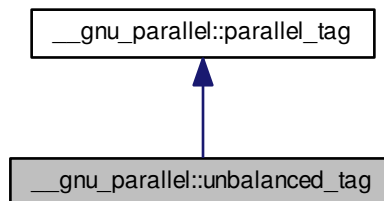
Definition at line 42 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.181 `__gnu_parallel::unbalanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



Public Member Functions

- [_ThreadIndex](#) [__get_num_threads](#) ()
- void [set_num_threads](#) ([_ThreadIndex](#) `__num_threads`)

4.181.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

4.181.2 Member Function Documentation

4.181.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::get_num_threads ()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::parallel_sort()`.

4.181.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

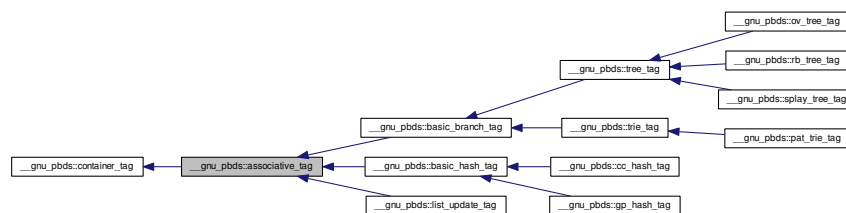
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.182 `__gnu_pbds::associative_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::associative_tag`:



4.182.1 Detailed Description

Basic associative-container.

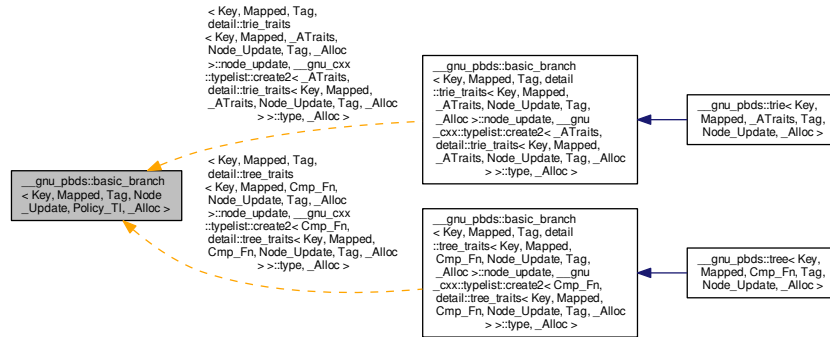
Definition at line 135 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.183 `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`:



Public Types

- typedef `Node_Update` **node_update**

Protected Member Functions

- **basic_branch** (const [basic_branch](#) &other)
- template<typename T0 >
basic_branch (T0 t0)
- template<typename T0 , typename T1 >
basic_branch (T0 t0, T1 t1)
- template<typename T0 , typename T1 , typename T2 >
basic_branch (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 , typename T2 , typename T3 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

4.183.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>class `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`

A branched, tree-like (tree, trie) container abstraction.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Node_Update</i>	Updates nodes, restores invariants.
<i>Policy_TL</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `tree_tag`, `trie_tag`, and their descendants.

Base choices are: `detail::ov_tree_map`, `detail::rb_tree_map`, `detail::splay_tree_map`, and `detail::pat_trie_map`.

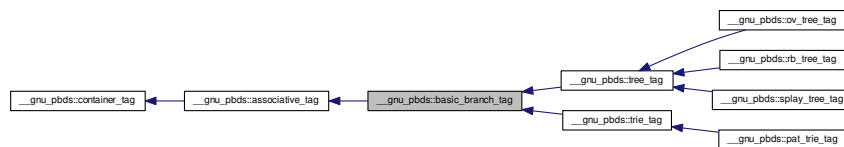
Definition at line 555 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.184 `__gnu_pbds::basic_branch_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_branch_tag`:



4.184.1 Detailed Description

Basic branch structure.

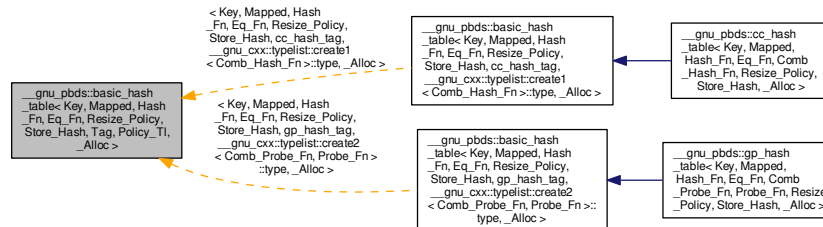
Definition at line 147 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.185 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`:



Protected Member Functions

- **basic_hash_table** (const [basic_hash_table](#) &other)
- template<typename T0 >
basic_hash_table (T0 t0)
- template<typename T0 , typename T1 >
basic_hash_table (T0 t0, T1 t1)
- template<typename T0 , typename T1 , typename T2 >
basic_hash_table (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 , typename T2 , typename T3 >
basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >
basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >
basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >
basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >
basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >
basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

4.185.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_Tl, typename _Alloc>class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`

A hashed container abstraction.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Policy_TL</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `cc_hash_tag`, `gp_hash_tag`, and descendants of `basic_hash_tag`.

Base choices are: `detail::cc_ht_map`, `detail::gp_ht_map`

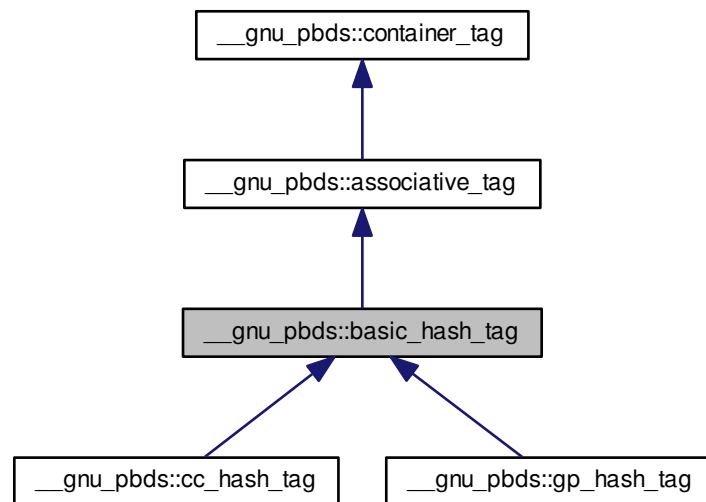
Definition at line 104 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.186 `__gnu_pbds::basic_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



4.186.1 Detailed Description

Basic hash structure.

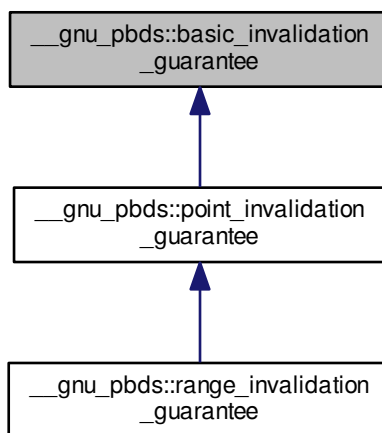
Definition at line 138 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.187 `__gnu_pbds::basic_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



4.187.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

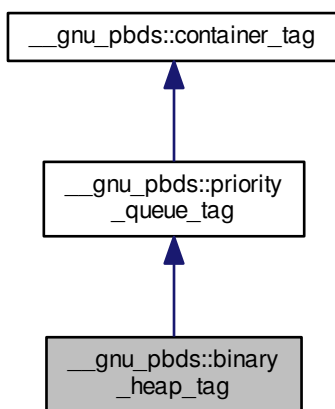
Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.188 `__gnu_pbds::binary_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



4.188.1 Detailed Description

Binary-heap (array-based).

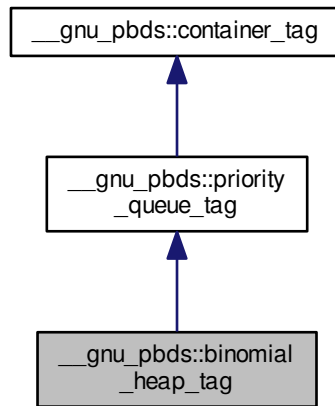
Definition at line 183 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.189 `__gnu_pbds::binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



4.189.1 Detailed Description

Binomial-heap.

Definition at line 177 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.190 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

Public Types

- enum { [external_load_access](#) }
- typedef `Size_Type` **size_type**

Public Member Functions

- [cc_hash_max_collision_check_resize_trigger](#) (float load=0.5)
- float [get_load](#) () const
- void [set_load](#) (float load)
- void **swap** ([cc_hash_max_collision_check_resize_trigger](#)< `External_Load_Access`, `Size_Type` > &other)

Protected Member Functions

- bool `is_grow_needed` (size_type size, size_type num_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (size_type num_entries)
- void `notify_externally_resized` (size_type new_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size_type num_entries)
- void `notify_resized` (size_type new_size)

4.190.1 Detailed Description

`template<bool External_Load_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.
 Definition at line 293 of file `hash_policy.hpp`.

4.190.2 Member Enumeration Documentation

4.190.2.1 `template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum`

Enumerator:

`external_load_access` Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 298 of file `hash_policy.hpp`.

4.190.3 Constructor & Destructor Documentation

4.190.3.1 `template<bool External_Load_Access, typename Size_Type > __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_hash_max_collision_check_resize_trigger (float load = 0.5)`

Default constructor, or constructor taking load, a `__load` factor which it will attempt to maintain.

Definition at line 44 of file `hash_policy.hpp`.

4.190.4 Member Function Documentation

4.190.4.1 `template<bool External_Load_Access, typename Size_Type > float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::get_load () const`
[inline]

Returns the current load.

Definition at line 190 of file `hash_policy.hpp`.

4.190.4.2 `template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::is_grow_needed (size_type size, size_type num_entries) const`
[inline], [protected]

Queries whether a grow is needed. This method is called only if this object indicated is needed.

Definition at line 133 of file `hash_policy.hpp`.

4.190.4.3 `template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::is_resize_needed () const` [inline], [protected]

Queries whether a resize is needed.

Definition at line 127 of file `hash_policy.hpp`.

4.190.4.4 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ()`
[protected]

Notifies the table was cleared.

Definition at line 121 of file `hash_policy.hpp`.

4.190.4.5 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_collision ()` [inline], [protected]

Notifies a search encountered a collision.

Definition at line 97 of file `hash_policy.hpp`.

4.190.4.6 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_end ()` [inline], [protected]

Notifies a search ended.

Definition at line 103 of file `hash_policy.hpp`.

4.190.4.7 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_start ()` [inline], [protected]

Notifies an erase search started.

Definition at line 91 of file `hash_policy.hpp`.

4.190.4.8 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erased (size_type num_entries) [inline], [protected]`

Notifies an element was erased.

Definition at line 115 of file hash_policy.hpp.

4.190.4.9 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_externally_resized (size_type new_size) [protected]`

Notifies the table was resized externally.

Definition at line 172 of file hash_policy.hpp.

4.190.4.10 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_collision () [inline], [protected]`

Notifies a search encountered a collision.

Definition at line 61 of file hash_policy.hpp.

4.190.4.11 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_end () [inline], [protected]`

Notifies a search ended.

Definition at line 67 of file hash_policy.hpp.

4.190.4.12 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_start () [inline], [protected]`

Notifies a find search started.

Definition at line 55 of file hash_policy.hpp.

4.190.4.13 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_collision () [inline], [protected]`

Notifies a search encountered a collision.

Definition at line 79 of file hash_policy.hpp.

4.190.4.14 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_end () [inline], [protected]`

Notifies a search ended.

Definition at line 85 of file hash_policy.hpp.

4.190.4.15 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_start () [inline], [protected]`

Notifies an insert search started.

Definition at line 73 of file hash_policy.hpp.

4.190.4.16 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted (size_type num_entries) [inline], [protected]`

Notifies an element was inserted.

Definition at line 109 of file hash_policy.hpp.

4.190.4.17 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized (size_type new_size) [protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 139 of file hash_policy.hpp.

4.190.4.18 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::set_load (float load)`

Sets the load; does not resize the container.

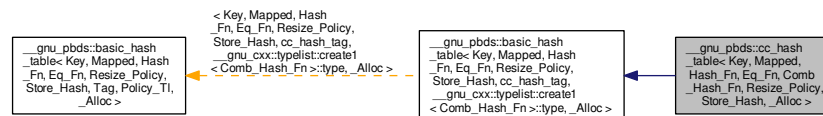
Definition at line 205 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- [hash_policy.hpp](#)
- [cc_hash_max_collision_check_resize_trigger_imp.hpp](#)

4.191 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`:



Public Types

- typedef `Comb_Hash_Fn` **comb_hash_fn**
- typedef `cc_hash_tag` **container_category**

- typedef `Eq_Fn` **eq_fn**
- typedef `Hash_Fn` **hash_fn**
- typedef `Resize_Policy` **resize_policy**

Public Member Functions

- [cc_hash_table](#) ()
- [cc_hash_table](#) (const hash_fn &h)
- [cc_hash_table](#) (const hash_fn &h, const eq_fn &e)
- [cc_hash_table](#) (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- [cc_hash_table](#) (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- template<typename It >
[cc_hash_table](#) (It first, It last)
- template<typename It >
[cc_hash_table](#) (It first, It last, const hash_fn &h)
- template<typename It >
[cc_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It >
[cc_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- template<typename It >
[cc_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- **cc_hash_table** (const [cc_hash_table](#) &other)
- [cc_hash_table](#) & **operator=** (const [cc_hash_table](#) &other)
- void **swap** ([cc_hash_table](#) &other)

4.191.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>>class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`

A collision-chaining hash-based associative container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Hash_Fn</i>	Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If <i>Hash_Fn</i> is null_type, then the container will not compile if this value is true
<i>_Alloc</i>	Allocator type.

Base tag choices are: `cc_hash_tag`.

Base is `basic_hash_table`.

Definition at line 204 of file `assoc_container.hpp`.

4.191.2 Constructor & Destructor Documentation

4.191.2.1 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table () [inline]`

Default constructor.

Definition at line 217 of file `assoc_container.hpp`.

4.191.2.2 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `Hash_Fn` object of the container object.

Definition at line 221 of file `assoc_container.hpp`.

4.191.2.3 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h, const eq_fn & e) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 228 of file `assoc_container.hpp`.

4.191.2.4 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 236 of file `assoc_container.hpp`.

```
4.191.2.5 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash =
detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped,
Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( const hash_fn & h, const eq_fn
& e, const comb_hash_fn & ch, const resize_policy & rp ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 245 of file `assoc_container.hpp`.

```
4.191.2.6 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb-
_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
>::cc_hash_table ( It first, It last ) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 253 of file `assoc_container.hpp`.

```
4.191.2.7 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb-
_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
>::cc_hash_table ( It first, It last, const hash_fn & h ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 260 of file `assoc_container.hpp`.

```
4.191.2.8 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb-
_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
>::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 271 of file `assoc_container.hpp`.

```
4.191.2.9  template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb-
            _hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
            Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
            __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
            >::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 283 of file `assoc_container.hpp`.

```
4.191.2.10 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
            typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb-
            _hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
            Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
            __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
            >::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch, const resize_policy
            & rp ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

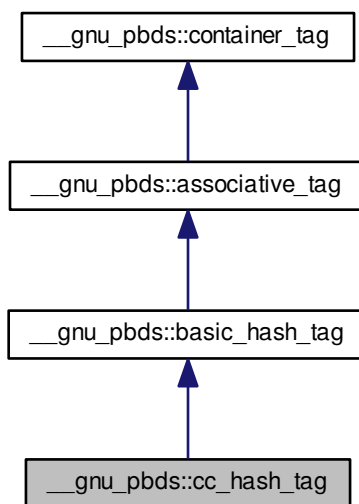
Definition at line 297 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.192 __gnu_pbds::cc_hash_tag Struct Reference

Inheritance diagram for __gnu_pbds::cc_hash_tag:



4.192.1 Detailed Description

Collision-chaining hash.

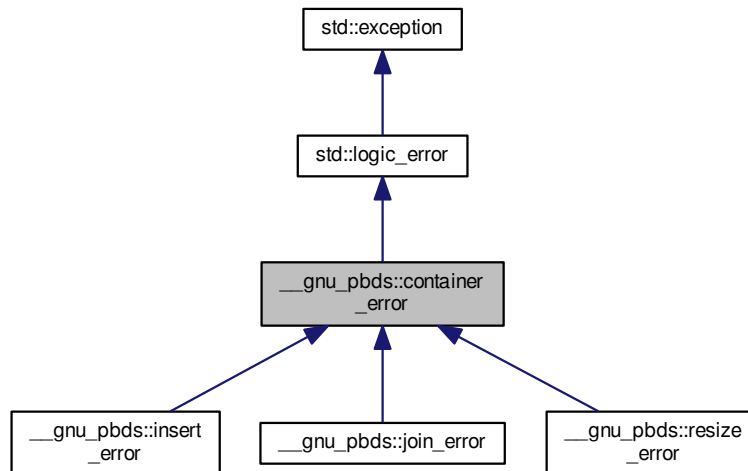
Definition at line 141 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.193 `__gnu_pbds::container_error` Struct Reference

Inheritance diagram for `__gnu_pbds::container_error`:



Public Member Functions

- virtual const char * [what](#) () const noexcept

4.193.1 Detailed Description

Base class for exceptions.

Definition at line 57 of file `exception.hpp`.

4.193.2 Member Function Documentation

4.193.2.1 virtual const char* `std::logic_error::what` () const [virtual],[noexcept],[inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

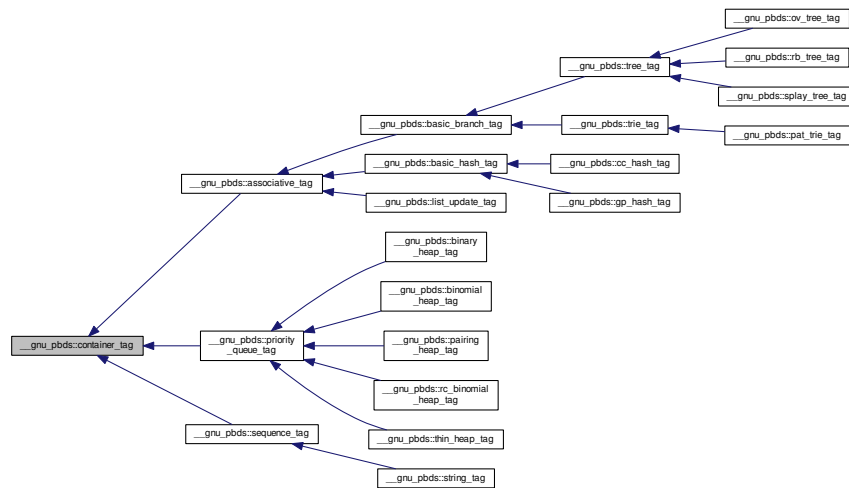
Reimplemented in [std::future_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.194 `__gnu_pbds::container_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::container_tag`:



4.194.1 Detailed Description

Base data structure tag.

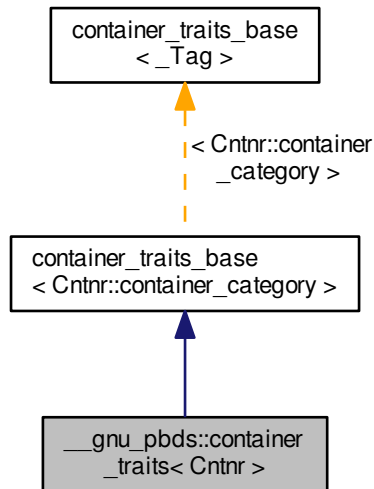
Definition at line 125 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.195 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::container_traits< Cntnr >`:



Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `container_traits_base< container_category >` **base_type**
- typedef `Cntnr::container_category` **container_category**
- typedef `Cntnr` **container_type**
- typedef `base_type::invalidation_guarantee` **invalidation_guarantee**

4.195.1 Detailed Description

```
template<typename Cntnr>struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

Definition at line 418 of file `tag_and_trait.hpp`.

4.195.2 Member Enumeration Documentation

4.195.2.1 `template<typename Cntnr >` anonymous enum

Enumerator:

`order_preserving` True only if `Cntnr` objects guarantee storing keys by order.

erase_can_throw True only if erasing a key can throw.

split_join_can_throw True only if split or join operations can throw.

reverse_iteration True only reverse iterators are supported.

Definition at line 426 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.196 `__gnu_pbds::container_traits_base< binary_heap_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [binary_heap_tag](#) **container_category**
- typedef [basic_invalidation_guarantee](#) **invalidation_guarantee**

4.196.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< binary_heap_tag >`

Specialization, binary heap.

Definition at line 400 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.197 `__gnu_pbds::container_traits_base< binomial_heap_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [binomial_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.197.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< binomial_heap_tag >`

Specialization, binomial heap.

Definition at line 368 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.198 `__gnu_pbds::container_traits_base< cc_hash_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [cc_hash_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.198.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< cc_hash_tag >`

Specialization, cc hash.

Definition at line 224 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.199 `__gnu_pbds::container_traits_base< gp_hash_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [gp_hash_tag](#) **container_category**
- typedef [basic_invalidation_guarantee](#) **invalidation_guarantee**

4.199.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< gp_hash_tag >`

Specialization, gp hash.

Definition at line 240 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.200 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [list_update_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.200.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< list_update_tag >`

Specialization, list update.

Definition at line 320 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.201 `__gnu_pbds::container_traits_base< ov_tree_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [ov_tree_tag](#) **container_category**
- typedef [basic_invalidation_guarantee](#) **invalidation_guarantee**

4.201.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< ov_tree_tag >`

Specialization, ov tree.

Definition at line 288 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.202 `__gnu_pbds::container_traits_base< pairing_heap_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [pairing_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.202.1 Detailed Description

`template<> struct __gnu_pbds::container_traits_base< pairing_heap_tag >`

Specialization, pairing heap.

Definition at line 336 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.203 `__gnu_pbds::container_traits_base< pat_trie_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [pat_trie_tag](#) **container_category**
- typedef [range_invalidation_guarantee](#) **invalidation_guarantee**

4.203.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< pat_trie_tag >`

Specialization, pat trie.

Definition at line 304 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.204 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [rb_tree_tag](#) **container_category**
- typedef [range_invalidation_guarantee](#) **invalidation_guarantee**

4.204.1 Detailed Description

`template<>struct __gnu_pbds::container_traits_base< rb_tree_tag >`

Specialization, rb tree.

Definition at line 256 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.205 `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [rc_binomial_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.205.1 Detailed Description

```
template<> struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
```

Specialization, rc binomial heap.

Definition at line 384 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.206 `__gnu_pbds::container_traits_base< splay_tree_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [splay_tree_tag](#) **container_category**
- typedef [range_invalidation_guarantee](#) **invalidation_guarantee**

4.206.1 Detailed Description

```
template<> struct __gnu_pbds::container_traits_base< splay_tree_tag >
```

Specialization, splay tree.

Definition at line 272 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.207 `__gnu_pbds::container_traits_base< thin_heap_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [thin_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.207.1 Detailed Description

```
template<> struct __gnu_pbds::container_traits_base< thin_heap_tag >
```

Specialization, thin heap.

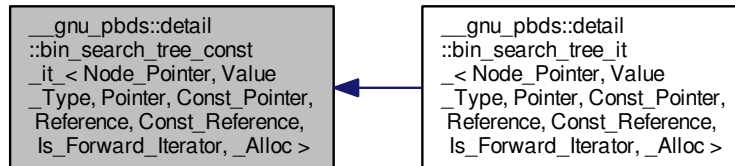
Definition at line 352 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.208 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `std::bidirectional_iterator_tag` **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_const_it_** (const Node_Pointer p_nd=0)
- **bin_search_tree_const_it_** (const [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) &other)
- bool **operator!=** (const [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) &other) const
- bool **operator!=** (const [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) &other) const
- const_reference **operator*** () const
- [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) & **operator++** ()
- [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) **operator++** (int)

- [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- const_pointer **operator->** () const
- [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

4.208.1 Detailed Description

`template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_const_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`

Const iterator.

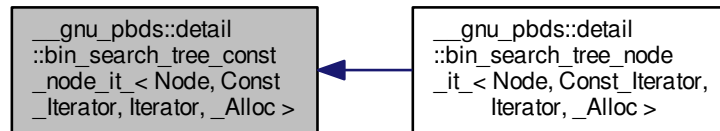
Definition at line 105 of file `point_iterators.hpp`.

The documentation for this class was generated from the following file:

- [point_iterators.hpp](#)

4.209 `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



Public Types

- typedef Const_Iterator [const_reference](#)
- typedef [trivial_iterator_difference_type](#) difference_type
- typedef [trivial_iterator_tag](#) iterator_category
- typedef _Alloc::template rebind< [metadata_type](#) > ::other::const_reference [metadata_const_reference](#)
- typedef Node::metadata_type [metadata_type](#)
- typedef Const_Iterator [reference](#)
- typedef Const_Iterator [value_type](#)

Public Member Functions

- **bin_search_tree_const_node_it_** (const node_pointer p_nd=0)
- [bin_search_tree_const_node_it_](#) < Node, Const_Iterator, Iterator, _Alloc > [get_l_child](#) () const
- [metadata_const_reference](#) [get_metadata](#) () const
- [bin_search_tree_const_node_it_](#) < Node, Const_Iterator, Iterator, _Alloc > [get_r_child](#) () const
- bool [operator!=](#) (const [bin_search_tree_const_node_it_](#) < Node, Const_Iterator, Iterator, _Alloc > &other) const
- [const_reference](#) [operator*](#) () const
- bool [operator==](#) (const [bin_search_tree_const_node_it_](#) < Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

4.209.1 Detailed Description

`template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >`

Const node iterator.

Definition at line 58 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2 Member Typedef Documentation

4.209.2.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::const_reference`

Iterator's `__const` reference type.

Definition at line 80 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef
trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,
Const_Iterator, Iterator, _Alloc >::difference_type`

Difference type.

Definition at line 71 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::iterator_category`

Category.

Definition at line 68 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template
rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference`

Const metadata reference type.

Definition at line 88 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::metadata_type`

Metadata type.

Definition at line 83 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference`

Iterator's reference type.

Definition at line 77 of file `bin_search_tree_/node_iterators.hpp`.

4.209.2.7 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator
 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type`

Iterator's value type.

Definition at line 74 of file `bin_search_tree_/node_iterators.hpp`.

4.209.3 Member Function Documentation

4.209.3.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_-
 it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,
 Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]`

Returns the `__const` node iterator associated with the left node.

Definition at line 107 of file `bin_search_tree_/node_iterators.hpp`.

4.209.3.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference
 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata () const [inline]`

Metadata access.

Definition at line 102 of file `bin_search_tree_/node_iterators.hpp`.

4.209.3.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_-
 it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,
 Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]`

Returns the `__const` node iterator associated with the right node.

Definition at line 112 of file `bin_search_tree_/node_iterators.hpp`.

4.209.3.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds-
 ::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (const
 bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

4.209.3.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > const_reference
 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* ()
 const [inline]`

Access.

Definition at line 97 of file `bin_search_tree_/node_iterators.hpp`.

4.209.3.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds-
 ::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator==(const
 bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline]`

Compares to a different iterator object.

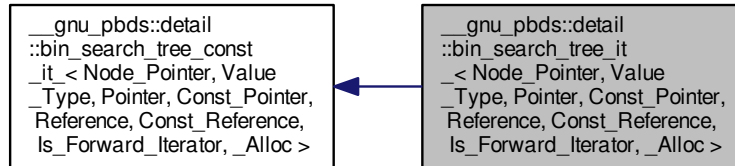
Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin_search_tree_/node_iterators.hpp](#)

4.210 `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_it_** (const Node_Pointer p_nd=0)
- **bin_search_tree_it_** (const [bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) &other)
- bool **operator!=** (const [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) &other) const
- bool **operator!=** (const [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) &other) const
- [bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) **operator*** () const
- [bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#) & **operator++** ()

- [bin_search_tree_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- [bin_search_tree_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- [bin_search_tree_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- [bin_search_tree_const_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > ::pointer **operator->** () const
- [bin_search_tree_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- [bin_search_tree_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Types

- typedef
[bin_search_tree_const_it](#)
< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **base_it_type**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

4.210.1 Detailed Description

template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>class `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`

Iterator.

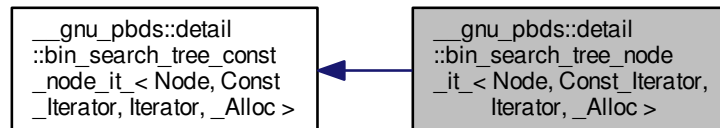
Definition at line 282 of file `point_iterators.hpp`.

The documentation for this class was generated from the following file:

- [point_iterators.hpp](#)

4.211 `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



Public Types

- typedef Iterator [const_reference](#)
- typedef [trivial_iterator_difference_type](#) difference_type
- typedef [trivial_iterator_tag](#) iterator_category
- typedef `_Alloc::template rebind< metadata_type >` `::other::const_reference` [metadata_const_reference](#)
- typedef `Node::metadata_type` [metadata_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value_type](#)

Public Member Functions

- **bin_search_tree_node_it_** (const node_pointer p_nd=0)
- **bin_search_tree_node_it_**< Node, Const_Iterator, Iterator, _Alloc > **get_l_child** () const
- **metadata_const_reference** **get_metadata** () const
- **bin_search_tree_node_it_**< Node, Const_Iterator, Iterator, _Alloc > **get_r_child** () const
- bool **operator!=** (const **bin_search_tree_const_node_it_**< Node, Const_Iterator, Iterator, _Alloc > &other) const
- Iterator **operator*** () const
- bool **operator==** (const **bin_search_tree_const_node_it_**< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

4.211.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Node iterator.

Definition at line 136 of file bin_search_tree_/node_iterators.hpp.

4.211.2 Member Typedef Documentation

4.211.2.1 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::const_reference

Iterator's __const reference type.

Definition at line 153 of file bin_search_tree_/node_iterators.hpp.

4.211.2.2 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef
trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,
Const_Iterator, Iterator, _Alloc >::difference_type [inherited]

Difference type.

Definition at line 71 of file bin_search_tree_/node_iterators.hpp.

4.211.2.3 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::iterator_category [inherited]

Category.

Definition at line 68 of file bin_search_tree_/node_iterators.hpp.

4.211.2.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template
rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]`

Const metadata reference type.

Definition at line 88 of file bin_search_tree_/node_iterators.hpp.

4.211.2.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::metadata_type [inherited]`

Metadata type.

Definition at line 83 of file bin_search_tree_/node_iterators.hpp.

4.211.2.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference`

Iterator's reference type.

Definition at line 150 of file bin_search_tree_/node_iterators.hpp.

4.211.2.7 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type`

Iterator's value type.

Definition at line 147 of file bin_search_tree_/node_iterators.hpp.

4.211.3 Member Function Documentation

4.211.3.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_<Node,
Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::get_l_child () const [inline]`

Returns the node iterator associated with the left node.

Definition at line 167 of file bin_search_tree_/node_iterators.hpp.

4.211.3.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata (
) const [inline], [inherited]`

Metadata access.

Definition at line 102 of file bin_search_tree_/node_iterators.hpp.

4.211.3.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_<Node,
Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::get_r_child () const [inline]`

Returns the node iterator associated with the right node.

Definition at line 175 of file bin_search_tree_/node_iterators.hpp.

4.211.3.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline],
[inherited]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

4.211.3.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > Iterator
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* () const
[inline]`

Access.

Definition at line 162 of file `bin_search_tree_/node_iterators.hpp`.

4.211.3.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator== (
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline],
[inherited]`

Compares to a different iterator object.

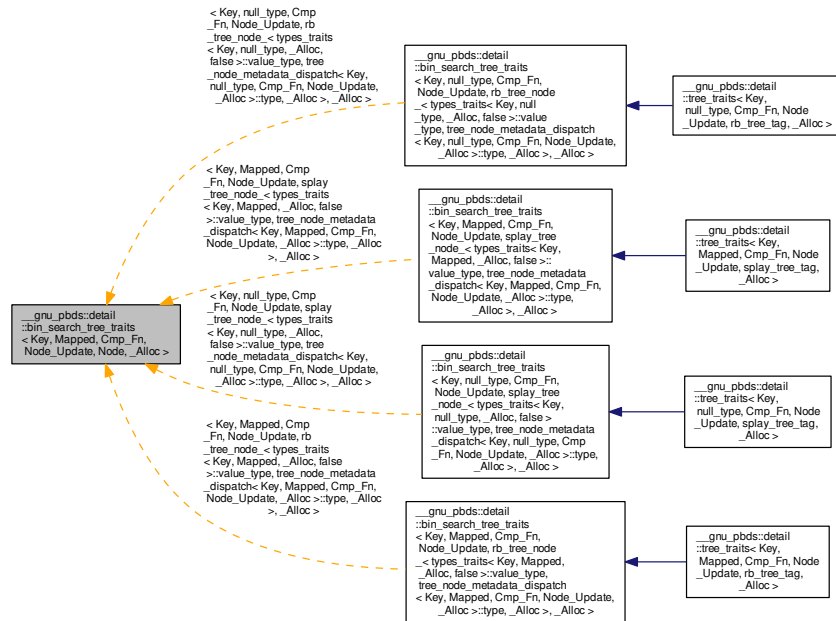
Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin_search_tree_/node_iterators.hpp](#)

4.212 `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`:



Public Types

- typedef `bin_search_tree_const_it`
`< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > const_reverse_iterator`
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it`
`< Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator`
- typedef

```

    bin_search_tree_node_it_< Node,
    point_const_iterator,
    point_iterator, _Alloc > node_iterator
• typedef Node_Update
  < node\_const\_iterator,
  node\_iterator, Cmp_Fn, _Alloc > node_update
• typedef
  \_\_gnu\_pbds::null\_node\_update
  < node\_const\_iterator,
  node\_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer
• typedef
  bin\_search\_tree\_const\_it\_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_const_iterator
• typedef bin\_search\_tree\_it\_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_iterator
• typedef bin\_search\_tree\_it\_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  false, _Alloc > reverse_iterator

```

4.212.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename
_Alloc > class Node_Update, class Node, typename _Alloc> struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn,
Node_Update, Node, _Alloc >
```

Binary search tree traits, primary template.

Definition at line 63 of file `bin_search_tree_/traits.hpp`.

4.212.2 Member Typedef Documentation

4.212.2.1 `template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc> typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin_search_tree_/traits.hpp](#)

4.213 `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

Public Types

- typedef [bin_search_tree_const_it_](#)
`< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > const_reverse_iterator`
- typedef Node **node**
- typedef [bin_search_tree_const_node_it_](#)
`< Node, point_const_iterator,
point_iterator, _Alloc > node_const_iterator`
- typedef [node_const_iterator](#) **node_iterator**
- typedef Node_Update
`< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > node_update`
- typedef [__gnu_pbds::null_node_update](#)
`< node_const_iterator,`

- `node_iterator`, `Cmp_Fn`, `_Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_`
`< typename _Alloc::template`
`rebind< node >::other::pointer,`
`typename`
`type_traits::value_type,`
`typename type_traits::pointer,`
`typename`
`type_traits::const_pointer,`
`typename`
`type_traits::reference,`
`typename`
`type_traits::const_reference,`
`true, _Alloc > point_const_iterator`
- typedef `point_const_iterator` `point_iterator`
- typedef `const_reverse_iterator` `reverse_iterator`

4.213.1 Detailed Description

`template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >`

Specialization.

Definition at line 169 of file `bin_search_tree_/traits.hpp`.

4.213.2 Member Typedef Documentation

4.213.2.1 `template<typename Key , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node , typename _Alloc > typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

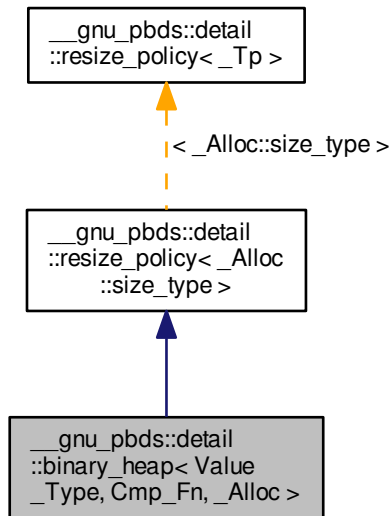
Definition at line 221 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin_search_tree_/traits.hpp](#)

4.214 `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `cond_dealtor`
`< value_type, _Alloc >` **cond_dealtor_t**
- typedef
`binary_heap_const_iterator_`
`< value_type, entry,`
`simple_value, _Alloc >` **const_iterator**
- typedef
`value_allocator::const_pointer` **const_pointer**
- typedef
`value_allocator::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `__conditional_type`
`< simple_value, value_type,`
`pointer >::__type` **entry**
- typedef `_Alloc::template`
`rebind< entry >::other` **entry_allocator**
- typedef `entry_cmp< Value_Type,`
`Cmp_Fn, _Alloc, is_simple`
`< Value_Type >::value >::type` **entry_cmp**
- typedef `entry_allocator::pointer` **entry_pointer**

- typedef `const_iterator` `iterator`
- typedef
`binary_heap_point_const_iterator_`
`< value_type, entry,`
`simple_value, _Alloc >` `point_const_iterator`
- typedef `point_const_iterator` `point_iterator`
- typedef `value_allocator::pointer` `pointer`
- typedef `value_allocator::reference` `reference`
- typedef
`__gnu_pbds::detail::resize_policy`
`< typename _Alloc::size_type >` `resize_policy`
- typedef `_Alloc::size_type` `size_type`
- typedef `Value_Type` `value_type`

Public Member Functions

- `binary_heap` (`const cmp_fn &`)
- `binary_heap` (`const binary_heap &`)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `void clear` ()
- `bool empty` () `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `void erase` (`point_iterator`)
- `void erase_at` (`entry_pointer`, `size_type`, `false_type`)
- `void erase_at` (`entry_pointer`, `size_type`, `true_type`)
- `template<typename Pred >`
`size_type erase_if` (`Pred`)
- `Cmp_Fn & get_cmp_fn` ()
- `const Cmp_Fn & get_cmp_fn` () `const`
- `size_type get_new_size_for_arbitrary` (`size_type`) `const`
- `size_type get_new_size_for_grow` () `const`
- `size_type get_new_size_for_shrink` () `const`
- `bool grow_needed` (`size_type`) `const`
- `void join` (`binary_heap &`)
- `size_type max_size` () `const`
- `void modify` (`point_iterator`, `const_reference`)
- `void notify_arbitrary` (`size_type`)
- `void notify_grow_resize` ()
- `void notify_shrink_resize` ()
- `void pop` ()
- `point_iterator push` (`const_reference`)
- `bool resize_needed_for_grow` (`size_type`) `const`
- `bool resize_needed_for_shrink` (`size_type`) `const`
- `bool shrink_needed` (`size_type`) `const`
- `size_type size` () `const`
- `template<typename Pred >`
`void split` (`Pred`, `binary_heap &`)
- `void swap` (`resize_policy< _Alloc::size_type > &`)
- `void swap` (`binary_heap &`)
- `const_reference top` () `const`

Static Public Attributes

- static const `_Alloc::size_type` **min_size**

Protected Member Functions

- template<typename It >
void **copy_from_range** (It, It)

4.214.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class `__gnu_pbds::detail::binary_heap_< Value_Type, Cmp_Fn, _Alloc >`

Binary heaps composed of resize and compare policies.

Based on CLRS.

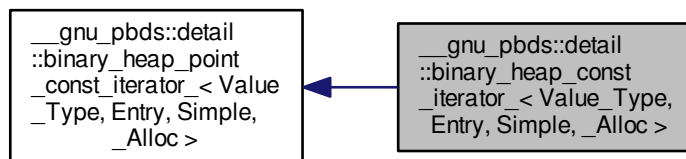
Definition at line 84 of file `binary_heap_.hpp`.

The documentation for this class was generated from the following files:

- [binary_heap_.hpp](#)
- [binary_heap_/insert_fn_imps.hpp](#)
- [binary_heap_/constructors_destructor_fn_imps.hpp](#)
- [binary_heap_/iterators_fn_imps.hpp](#)
- [binary_heap_/erase_fn_imps.hpp](#)
- [binary_heap_/info_fn_imps.hpp](#)
- [binary_heap_/find_fn_imps.hpp](#)
- [binary_heap_/split_join_fn_imps.hpp](#)
- [binary_heap_/policy_access_fn_imps.hpp](#)

4.215 `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

Public Member Functions

- `binary_heap_const_iterator_` (`entry_pointer p_e`)
- `binary_heap_const_iterator_` ()
- `binary_heap_const_iterator_` (`const binary_heap_const_iterator_ &other`)
- `bool operator!=` (`const binary_heap_const_iterator_ &other`) `const`
- `bool operator!=` (`const binary_heap_point_const_iterator_ &other`) `const`
- `const_reference operator*` () `const`
- `binary_heap_const_iterator_ & operator++` ()
- `binary_heap_const_iterator_ operator++` (`int`)
- `const_pointer operator->` () `const`
- `bool operator==` (`const binary_heap_const_iterator_ &other`) `const`
- `bool operator==` (`const binary_heap_point_const_iterator_ &other`) `const`

Public Attributes

- `entry_pointer m_p_e`

4.215.1 Detailed Description

`template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >`

Const point-type iterator.

Definition at line 60 of file `binary_heap_/const_iterator.hpp`.

4.215.2 Member Typedef Documentation

4.215.2.1 `template<typename Value_Type, typename Entry, bool Simple, typename _Alloc > typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_pointer`

Iterator's const pointer type.

Definition at line 80 of file `binary_heap_/const_iterator.hpp`.

4.215.2.2 `template<typename Value_Type, typename Entry, bool Simple, typename _Alloc > typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_reference`

Iterator's const reference type.

Definition at line 86 of file `binary_heap_/const_iterator.hpp`.

4.215.2.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::difference_type
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::difference_type`

Difference type.

Definition at line 71 of file `binary_heap_/const_iterator.hpp`.

4.215.2.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef std::forward_iterator_tag
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::iterator_category`

Category.

Definition at line 68 of file `binary_heap_/const_iterator.hpp`.

4.215.2.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::pointer
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::pointer`

Iterator's pointer type.

Definition at line 77 of file `binary_heap_/const_iterator.hpp`.

4.215.2.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::reference
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::reference`

Iterator's reference type.

Definition at line 83 of file `binary_heap_/const_iterator.hpp`.

4.215.2.7 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::value_type
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type`

Iterator's value type.

Definition at line 74 of file `binary_heap_/const_iterator.hpp`.

4.215.3 Constructor & Destructor Documentation

4.215.3.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail-
::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_()
[inline]`

Default constructor.

Definition at line 94 of file `binary_heap_/const_iterator.hpp`.

4.215.3.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary-
_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_(const
binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) [inline]`

Copy constructor.

Definition at line 99 of file `binary_heap_/const_iterator.hpp`.

4.215.4 Member Function Documentation

4.215.4.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=(const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 110 of file `binary_heap_/const_iterator.hpp`.

4.215.4.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=(const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline], [inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

4.215.4.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator*() const [inline], [inherited]`

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

4.215.4.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator-> () const [inline], [inherited]`

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

4.215.4.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==(const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content to a different iterator object.

Definition at line 105 of file `binary_heap_/const_iterator.hpp`.

4.215.4.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==(const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline], [inherited]`

Compares content to a different iterator object.

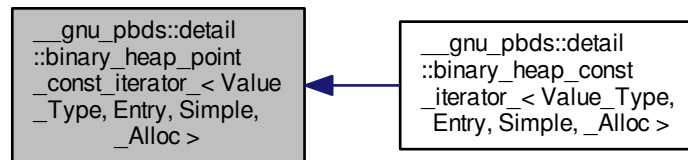
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary_heap_/const_iterator.hpp](#)

4.216 `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



Public Types

- typedef `_Alloc::template rebind< value_type > ::other::const_pointer` `const_pointer`
- typedef `_Alloc::template rebind< value_type > ::other::const_reference` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `_Alloc::template rebind< value_type > ::other::pointer` `pointer`
- typedef `_Alloc::template rebind< value_type > ::other::reference` `reference`
- typedef `Value_Type` `value_type`

Public Member Functions

- `binary_heap_point_const_iterator_ (entry_pointer p_e)`
- `binary_heap_point_const_iterator_ ()`
- `binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_ &other)`
- `bool operator!= (const binary_heap_point_const_iterator_ &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const binary_heap_point_const_iterator_ &other) const`

Public Attributes

- entry_pointer `m_p_e`

Protected Types

- typedef _Alloc::template
rebind< Entry >
::other::pointer **entry_pointer**

4.216.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

Definition at line 55 of file binary_heap_/point_const_iterator.hpp.

4.216.2 Member Typedef Documentation

4.216.2.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template
rebind<value_type>::other::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_pointer

Iterator's const pointer type.

Definition at line 77 of file binary_heap_/point_const_iterator.hpp.

4.216.2.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template
rebind<value_type>::other::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_reference

Iterator's const reference type.

Definition at line 87 of file binary_heap_/point_const_iterator.hpp.

4.216.2.3 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef
trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::difference_type

Difference type.

Definition at line 65 of file binary_heap_/point_const_iterator.hpp.

4.216.2.4 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef trivial_iterator_tag
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::iterator_category

Category.

Definition at line 62 of file binary_heap_/point_const_iterator.hpp.

4.216.2.5 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template
rebind<value_type>::other::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::pointer

Iterator's pointer type.

Definition at line 72 of file binary_heap_/point_const_iterator.hpp.

4.216.2.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template
rebind<value_type>::other::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::reference`

Iterator's reference type.

Definition at line 82 of file `binary_heap_/point_const_iterator.hpp`.

4.216.2.7 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef Value_Type
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type`

Iterator's value type.

Definition at line 68 of file `binary_heap_/point_const_iterator.hpp`.

4.216.3 Constructor & Destructor Documentation

4.216.3.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_
heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_ ()
[inline]`

Default constructor.

Definition at line 95 of file `binary_heap_/point_const_iterator.hpp`.

4.216.3.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_
heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_ (
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) [inline]`

Copy constructor.

Definition at line 99 of file `binary_heap_/point_const_iterator.hpp`.

4.216.4 Member Function Documentation

4.216.4.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds-
::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!= (const
binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

4.216.4.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator* ()
const [inline]`

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

4.216.4.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator-> (
) const [inline]`

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

4.216.4.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==(const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content to a different iterator object.

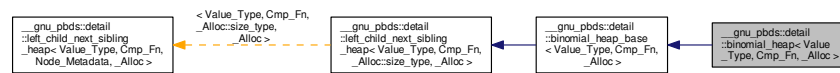
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary_heap_/point_const_iterator.hpp](#)

4.217 `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- `typedef base_type::allocator_type allocator_type`
- `typedef base_type::cmp_fn cmp_fn`
- `typedef base_type::const_iterator const_iterator`
- `typedef base_type::const_pointer const_pointer`
- `typedef base_type::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef base_type::iterator iterator`
- `typedef base_type::point_const_iterator point_const_iterator`
- `typedef base_type::point_iterator point_iterator`
- `typedef base_type::pointer pointer`
- `typedef base_type::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef Value_Type value_type`

Public Member Functions

- `binomial_heap (const Cmp_Fn &)`
- `binomial_heap (const binomial_heap &)`
- `iterator begin ()`
- `const_iterator begin () const`
- `void clear ()`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `void erase (point_iterator)`

- `template<typename Pred >`
`size_type erase_if (Pred)`
- `Cmp_Fn & get_cmp_fn ()`
- `const Cmp_Fn & get_cmp_fn () const`
- `void join (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)`
- `size_type max_size () const`
- `void modify (point_iterator, const_reference)`
- `void pop ()`
- `point_iterator push (const_reference)`
- `size_type size () const`
- `template<typename Pred >`
`void split (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)`
- `void swap (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)`
- `const_reference top () const`

Protected Types

- `typedef base_type::node node`
- `typedef _Alloc::template`
`rebind`
`< left_child_next_sibling_heap_node_`
`< Value_Type,`
`_Alloc::size_type, _Alloc >`
`>::other node_allocator`
- `typedef _Alloc::size_type node_metadata`
- `typedef std::pair`
`< node_pointer, node_pointer > node_pointer_pair`

Protected Member Functions

- `void actual_erase_node (node_pointer)`
- `void bubble_to_top (node_pointer)`
- `void clear_imp (node_pointer)`
- `template<typename It >`
`void copy_from_range (It, It)`
- `void find_max ()`
- `node_pointer get_new_node_for_insert (const_reference)`
- `node_pointer prune (Pred)`
- `void swap (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)`
- `void swap_with_parent (node_pointer, node_pointer)`
- `void to_linked_list ()`
- `void value_swap (left_child_next_sibling_heap &)`

Static Protected Member Functions

- `static void make_child_of (node_pointer, node_pointer)`
- `static node_pointer parent (node_pointer)`

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.217.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class `__gnu_pbds::detail::binomial_heap`< Value_Type, Cmp_Fn, _Alloc >

Binomial heap.

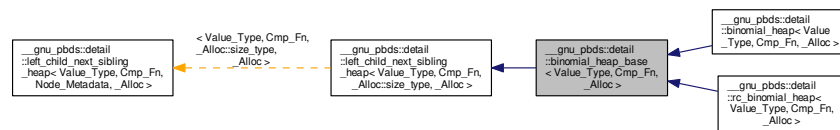
Definition at line 68 of file `binomial_heap.hpp`.

The documentation for this class was generated from the following files:

- [binomial_heap.hpp](#)
- [binomial_heap/constructors_destructor_fn_imps.hpp](#)

4.218 `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_v::const_pointer` **const_pointer**
- typedef `__rebind_v::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- `iterator begin ()`
- `const_iterator begin () const`
- `void clear ()`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `void erase (point_iterator)`
- `template<typename Pred >
size_type erase_if (Pred)`
- `Cmp_Fn & get_cmp_fn ()`
- `const Cmp_Fn & get_cmp_fn () const`
- `void join (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)`
- `size_type max_size () const`
- `void modify (point_iterator, const_reference)`
- `void pop ()`
- `point_iterator push (const_reference)`
- `size_type size () const`
- `template<typename Pred >
void split (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)`
- `void swap (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)`
- `const_reference top () const`

Protected Types

- `typedef base_type::node node`
- `typedef _Alloc::template
rebind
< left_child_next_sibling_heap_node_
< Value_Type,
_Alloc::size_type, _Alloc >
>::other node_allocator`
- `typedef
base_type::node_const_pointer node_const_pointer`
- `typedef _Alloc::size_type node_metadata`
- `typedef base_type::node_pointer node_pointer`
- `typedef std::pair
< node_pointer, node_pointer > node_pointer_pair`

Protected Member Functions

- `binomial_heap_base (const Cmp_Fn &)`
- `binomial_heap_base (const binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)`
- `void actual_erase_node (node_pointer)`
- `void bubble_to_top (node_pointer)`
- `void clear_imp (node_pointer)`
- `template<typename It >
void copy_from_range (It, It)`
- `void find_max ()`
- `node_pointer get_new_node_for_insert (const_reference)`

- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.218.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class `__gnu_pbds::detail::binomial_heap_base`< Value_Type, Cmp_Fn, _Alloc >

Base class for binomial heap.

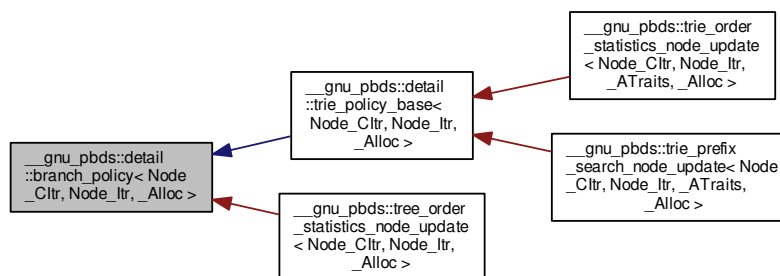
Definition at line 77 of file `binomial_heap_base.hpp`.

The documentation for this class was generated from the following files:

- [binomial_heap_base.hpp](#)
- [binomial_heap_base_/constructors_destructor_fn_imps.hpp](#)
- [binomial_heap_base_/find_fn_imps.hpp](#)
- [binomial_heap_base_/insert_fn_imps.hpp](#)
- [binomial_heap_base_/erase_fn_imps.hpp](#)
- [binomial_heap_base_/split_join_fn_imps.hpp](#)

4.219 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`:



Protected Types

- `typedef rebind_v::const_pointer` **const_pointer**
- `typedef rebind_v::const_reference` **const_reference**
- `typedef Node_Itr::value_type` **it_type**
- `typedef rebind_k::const_reference` **key_const_reference**
- `typedef value_type::first_type` **key_type**
- `typedef remove_const< key_type >::type` **rkey_type**
- `typedef remove_const< value_type >::type` **rcvalue_type**
- `typedef _Alloc::template rebind< rkey_type >::other` **rebind_k**
- `typedef _Alloc::template rebind< rcvalue_type >::other` **rebind_v**
- `typedef rebind_v::reference` **reference**
- `typedef std::iterator_traits< it_type >::value_type` **value_type**

Protected Member Functions

- virtual `it_type end ()=0`
- `it_type end_iterator () const`

Static Protected Member Functions

- static `key_const_reference extract_key (const_reference r_val)`

4.219.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename _Alloc>struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`

Primary template, base class for branch structure policies.

Definition at line 52 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch_policy.hpp](#)

4.220 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

Protected Types

- `typedef rebind_v::const_pointer` **const_pointer**
- `typedef rebind_v::const_reference` **const_reference**
- `typedef Node_Cltr::value_type` **it_type**
- `typedef rebind_v::const_reference` **key_const_reference**
- `typedef value_type` **key_type**

- typedef remove_const
< value_type >::type **rcvalue_type**
- typedef _Alloc::template
rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits
< it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** () const =0
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

4.220.1 Detailed Description

template<typename Node_Cltr, typename _Alloc>struct `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >`

Specialization for const iterators.

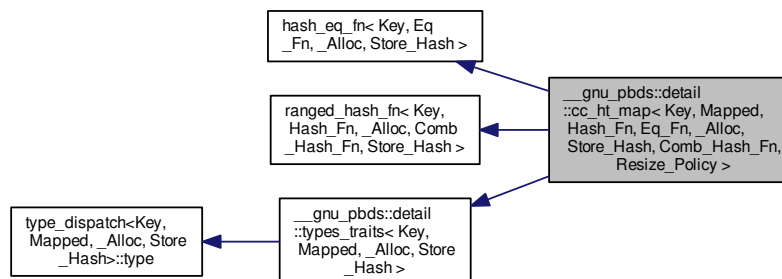
Definition at line 88 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch_policy.hpp](#)

4.221 `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`:



Public Types

- enum { **store_hash** }
- typedef `_Alloc` **allocator_type**
- typedef `Comb_Hash_Fn` **comb_hash_fn**
- typedef `const_iterator` **const_iterator**
- typedef `traits_base::const_pointer` **const_pointer**
- typedef `traits_base::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `Eq_Fn` **eq_fn**
- typedef `Hash_Fn` **hash_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key_const_pointer**
- typedef `traits_base::key_const_reference` **key_const_reference**
- typedef `traits_base::key_pointer` **key_pointer**
- typedef `traits_base::key_reference` **key_reference**
- typedef `traits_base::key_type` **key_type**
- typedef `traits_base::mapped_const_pointer` **mapped_const_pointer**
- typedef `traits_base::mapped_const_reference` **mapped_const_reference**
- typedef `traits_base::mapped_pointer` **mapped_pointer**
- typedef `traits_base::mapped_reference` **mapped_reference**
- typedef `traits_base::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `point_const_iterator` **point_const_iterator**
- typedef `point_iterator` **point_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize_policy**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**
- typedef `traits_base::value_type` **value_type**

Public Member Functions

- **cc_ht_map** (`const Hash_Fn &`)
- **cc_ht_map** (`const Hash_Fn &, const Eq_Fn &`)
- **cc_ht_map** (`const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &`)
- **cc_ht_map** (`const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &, const Resize_Policy &`)
- **cc_ht_map** (`const cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &`)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()

- `template<typename It >`
`void copy_from_range (It, It)`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `bool erase (key_const_reference)`
- `template<typename Pred >`
`size_type erase_if (Pred)`
- `point_iterator find (key_const_reference)`
- `point_const_iterator find (key_const_reference) const`
- `point_iterator find_end ()`
- `point_const_iterator find_end () const`
- `Comb_Hash_Fn & get_comb_hash_fn ()`
- `const Comb_Hash_Fn & get_comb_hash_fn () const`
- `Eq_Fn & get_eq_fn ()`
- `const Eq_Fn & get_eq_fn () const`
- `Hash_Fn & get_hash_fn ()`
- `const Hash_Fn & get_hash_fn () const`
- `Resize_Policy & get_resize_policy ()`
- `const Resize_Policy & get_resize_policy () const`
- `void initialize ()`
- `std::pair< point_iterator, bool > insert (const_reference r_val)`
- `size_type max_size () const`
- `mapped_reference operator[] (key_const_reference r_key)`
- `size_type size () const`
- `void swap (cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)`

Public Attributes

- no_throw_indicator `m_no_throw_copies_indicator`
- store_extra `m_store_extra_indicator`

Friends

- class `const_iterator_`
- class `iterator_`

4.221.1 Detailed Description

`template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy> class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`

A collision-chaining hash-based container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to<Key></code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Hash_Fn</i>	Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default <code>direct_mask_range_hashing</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_hash_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

Definition at line 139 of file `cc_ht_map.hpp`.

4.221.2 Member Enumeration Documentation

4.221.2.1 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy> anonymous enum`

Value stores hash, true or false.

Definition at line 200 of file `cc_ht_map.hpp`.

4.221.3 Member Function Documentation

4.221.3.1 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::empty () const` `[inline]`

True if `size() == 0`.

Definition at line 52 of file `cc_ht_map.hpp`.

4.221.3.2 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()`

Return current `comb_hash_fn`.

Definition at line 70 of file `cc_ht_map.hpp`.

4.221.3.3 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const`

Return current `const comb_hash_fn`.

Definition at line 76 of file `cc_ht_map.hpp`.

4.221.3.4 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ()`

Return current eq_fn.

Definition at line 58 of file `cc_ht_map.hpp`.

4.221.3.5 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const`

Return current const eq_fn.

Definition at line 64 of file `cc_ht_map.hpp`.

4.221.3.6 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn ()`

Return current hash_fn.

Definition at line 46 of file `cc_ht_map.hpp`.

4.221.3.7 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const`

Return current const hash_fn.

Definition at line 52 of file `cc_ht_map.hpp`.

4.221.3.8 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()`

Return current resize_policy.

Definition at line 82 of file `cc_ht_map.hpp`.

4.221.3.9 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const`

Return current const resize_policy.

Definition at line 88 of file `cc_ht_map.hpp`.

The documentation for this class was generated from the following files:

- [cc_ht_map.hpp](#)
- [cc_hash_table_map_/constructor_destructor_fn_imps.hpp](#)
- [entry_list_fn_imps.hpp](#)
- [cc_hash_table_map_/find_fn_imps.hpp](#)
- [cc_hash_table_map_/resize_fn_imps.hpp](#)
- [cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp](#)
- [cc_hash_table_map_/resize_store_hash_fn_imps.hpp](#)
- [size_fn_imps.hpp](#)
- [cc_hash_table_map_/policy_access_fn_imps.hpp](#)

- [cc_hash_table_map_/erase_fn_imps.hpp](#)
- [cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp](#)
- [cc_hash_table_map_/erase_store_hash_fn_imps.hpp](#)
- [cc_hash_table_map_/iterators_fn_imps.hpp](#)
- [cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp](#)
- [cc_hash_table_map_/insert_store_hash_fn_imps.hpp](#)

4.222 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

Public Types

- typedef HT_Map::entry **entry**
- typedef HT_Map::entry_allocator **entry_allocator**
- typedef __rebind_e::other **entry_allocator**
- typedef entry_allocator::pointer **entry_pointer**
- typedef HT_Map::key_type **key_type**

Public Member Functions

- **cond_dealtor** (entry_allocator *p_a, entry *p_e)
- **cond_dealtor** (entry_pointer p_e)
- void **set_key_destruct** ()
- void **set_no_action** ()
- void **set_no_action_destructor** ()

Protected Attributes

- bool **m_key_destruct**
- entry_allocator *const **m_p_a**
- entry *const **m_p_e**

4.222.1 Detailed Description

`template<typename Entry, typename _Alloc>class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >`

Conditional deallocate constructor argument.

Conditional key destructor, `cc_hash`.

Definition at line 50 of file `cond_dealtor.hpp`.

The documentation for this class was generated from the following files:

- [cond_dealtor.hpp](#)
- [cond_key_dtor_entry_dealtor.hpp](#)

4.223 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>`

Struct Template Reference

Public Types

- typedef `binary_heap<_VTp, Cmp_Fn, _Alloc>` `type`

4.223.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>
```

Specialization for `binary_heap`.

Definition at line 95 of file `priority_queue_base_dispatch.hpp`.

4.223.2 Member Typedef Documentation

4.223.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>::type`

Dispatched type.

Definition at line 99 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `priority_queue_base_dispatch.hpp`

4.224 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>`

Struct Template Reference

Public Types

- typedef `binomial_heap<_VTp, Cmp_Fn, _Alloc>` `type`

4.224.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>
```

Specialization for `binomial_heap`.

Definition at line 77 of file `priority_queue_base_dispatch.hpp`.

4.224.2 Member Typedef Documentation

4.224.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>::type`

Dispatched type.

Definition at line 81 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.225 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>` Struct Template Reference

Public Types

- typedef `pairing_heap<_VTp, Cmp_Fn, _Alloc>` `type`

4.225.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>`

Specialization for `pairing_heap`.

Definition at line 68 of file `priority_queue_base_dispatch.hpp`.

4.225.2 Member Typedef Documentation

4.225.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>::type`

Dispatched type.

Definition at line 72 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.226 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type>` Struct Template Reference

Public Types

- typedef `rc_binomial_heap<_VTp, Cmp_Fn, _Alloc>` `type`

4.226.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type>`

Specialization for `rc_binary_heap`.

Definition at line 86 of file `priority_queue_base_dispatch.hpp`.

4.226.2 Member Typedef Documentation

4.226.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type`

Dispatched type.

Definition at line 90 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.227 `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >` Struct Template Reference

Public Types

- typedef [thin_heap](#)<_VTp, Cmp_Fn, _Alloc> [type](#)

4.227.1 Detailed Description

`template<typename _VTp, typename Cmp_Fn, typename _Alloc> struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >`

Specialization for `thin_heap`.

Definition at line 104 of file `priority_queue_base_dispatch.hpp`.

4.227.2 Member Typedef Documentation

4.227.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >::type`

Dispatched type.

Definition at line 108 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.228 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >` Struct Template Reference

Public Types

- typedef [cc_ht_map](#)< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> [type](#)

4.228.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >`

Specialization collision-chaining hash map.

Definition at line 258 of file `container_base_dispatch.hpp`.

4.228.2 Member Typedef Documentation

4.228.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_TI > typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >::type`

Dispatched type.

Definition at line 275 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.229 `__gnu_pbds::detail::container_base_dispatch`< Key, Mapped, `_Alloc`, `gp_hash_tag`, `Policy_TI` > Struct Template Reference

Public Types

- typedef `gp_ht_map`< Key, Mapped, at0t, at1t, `_Alloc`, at3t::value, at4t, at5t, at2t > `type`

4.229.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >`

Specialization general-probe hash map.

Definition at line 303 of file `container_base_dispatch.hpp`.

4.229.2 Member Typedef Documentation

4.229.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_TI > typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >::type`

Dispatched type.

Definition at line 322 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.230 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `lu_map< Key, Mapped, at0t, _Alloc, at1t >` `type`

4.230.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >`

Specialization for list-update map.

Definition at line 107 of file `container_base_dispatch.hpp`.

4.230.2 Member Typedef Documentation

4.230.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 118 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

4.231 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `ov_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

4.231.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >`

Specialization ordered-vector tree map.

Definition at line 227 of file `container_base_dispatch.hpp`.

4.231.2 Member Typedef Documentation

4.231.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_TI > typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >::type`

Dispatched type.

Definition at line 237 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.232 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >` Struct Template Reference

Public Types

- typedef [pat_trie_map](#)< Key, Mapped, at1t, _Alloc > **type**

4.232.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >`

Specialization for PATRICIA trie map.

Definition at line 139 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.233 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >` Struct Template Reference

Public Types

- typedef [rb_tree_map](#)< Key, Mapped, at0t, at1t, _Alloc > [type](#)

4.233.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >`

Specialization for R-B tree map.

Definition at line 165 of file `container_base_dispatch.hpp`.

4.233.2 Member Typedef Documentation

4.233.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef rb_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 175 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.234 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef [splay_tree_map](#)< Key, Mapped, at0t, at1t, _Alloc > [type](#)

4.234.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >`

Specialization splay tree map.

Definition at line 195 of file `container_base_dispatch.hpp`.

4.234.2 Member Typedef Documentation

4.234.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 206 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.235 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `cc_ht_set`< Key, [null_type](#), at0t, at1t, _Alloc, at3t::value, at4t, at2t > [type](#)

4.235.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type,
_Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash set.

Definition at line 280 of file `container_base_dispatch.hpp`.

4.235.2 Member Typedef Documentation

```
4.235.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc,
at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl
>::type
```

Dispatched type.

Definition at line 298 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.236 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `gp_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` `type`

4.236.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type,
_Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash set.

Definition at line 327 of file `container_base_dispatch.hpp`.

4.236.2 Member Typedef Documentation

```
4.236.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc,
at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag,
Policy_Tl >::type
```

Dispatched type.

Definition at line 347 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.237 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >` Struct Template Reference

Public Types

- `typedef lu_set< Key, null_type, at0t, _Alloc, at1t > type`

4.237.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_TI> struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >`

Specialization for list-update set.

Definition at line 123 of file `container_base_dispatch.hpp`.

4.237.2 Member Typedef Documentation

4.237.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >::type`

Dispatched type.

Definition at line 134 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.238 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >` Struct Template Reference

Public Types

- `typedef ov_tree_set< Key, null_type, at0t, at1t, _Alloc > type`

4.238.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_TI> struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >`

Specialization ordered-vector tree set.

Definition at line 242 of file `container_base_dispatch.hpp`.

4.238.2 Member Typedef Documentation

4.238.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >::type`

Dispatched type.

Definition at line 253 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.239 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

Public Types

- `typedef pat_trie_set< Key, null_type, at1t, _Alloc > type`

4.239.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >`

Specialization for PATRICIA trie set.

Definition at line 151 of file `container_base_dispatch.hpp`.

4.239.2 Member Typedef Documentation

4.239.2.1 `template<typename Key , typename _Alloc , typename Policy_Tl > typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 160 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.240 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- `typedef rb_tree_set< Key, null_type, at0t, at1t, _Alloc > type`

4.240.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >`

Specialization for R-B tree set.

Definition at line 180 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.241 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- `typedef splay_tree_set< Key, null_type, at0t, at1t, _Alloc > type`

4.241.1 Detailed Description

`template<typename Key, typename _Alloc, typename Policy_Tl> struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >`

Specialization splay tree set.

Definition at line 211 of file `container_base_dispatch.hpp`.

4.241.2 Member Typedef Documentation

4.241.2.1 `template<typename Key , typename _Alloc , typename Policy_Tl > typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 222 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.242 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

Public Types

- `typedef direct_mask_range_hashing type`

4.242.1 Detailed Description

Primary template, `default_comb_hash_fn`.

Definition at line 80 of file `standard_policies.hpp`.

4.242.2 Member Typedef Documentation

4.242.2.1 `typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type`

Dispatched type.

Definition at line 83 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.243 `__gnu_pbds::detail::default_eq_fn< Key >` Struct Template Reference

Public Types

- typedef `std::equal_to< Key >` [type](#)

4.243.1 Detailed Description

`template<typename Key>struct __gnu_pbds::detail::default_eq_fn< Key >`

Primary template, `default_eq_fn`.

Definition at line 67 of file `standard_policies.hpp`.

4.243.2 Member Typedef Documentation

4.243.2.1 `template<typename Key> typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type`

Dispatched type.

Definition at line 70 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.244 `__gnu_pbds::detail::default_hash_fn< Key >` Struct Template Reference

Public Types

- typedef `std::tr1::hash< Key >` [type](#)

4.244.1 Detailed Description

`template<typename Key>struct __gnu_pbds::detail::default_hash_fn< Key >`

Primary template, `default_hash_fn`.

Definition at line 59 of file `standard_policies.hpp`.

4.244.2 Member Typedef Documentation

4.244.2.1 `template<typename Key> typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type`

Dispatched type.

Definition at line 62 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.245 `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >` Struct Template Reference

Public Types

- typedef `cond_type::__type` [type](#)

4.245.1 Detailed Description

`template<typename Comb_Probe_Fn> struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >`

Primary template, `default_probe_fn`.

Definition at line 117 of file `standard_policies.hpp`.

4.245.2 Member Typedef Documentation

4.245.2.1 `template<typename Comb_Probe_Fn > typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type`

Dispatched type.

Definition at line 129 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.246 `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >` Struct Template Reference

Public Types

- typedef
[hash_standard_resize_policy](#)
< `size_policy_type`, [trigger](#),
`false`, `size_type` > [type](#)

4.246.1 Detailed Description

`template<typename Comb_Hash_Fn> struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >`

Primary template, `default_resize_policy`.

Definition at line 88 of file `standard_policies.hpp`.

4.246.2 Member Typedef Documentation

4.246.2.1 `template<typename Comb_Hash_Fn> typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >::type`

Dispatched type.

Definition at line 105 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.247 `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >` Struct Template Reference

Public Types

- typedef
[trie_string_access_traits](#)
< [string_type](#) > type

4.247.1 Detailed Description

`template<typename Char, typename Char_Traits> struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`

Partial specialization, `default_trie_access_traits`.

Definition at line 142 of file `standard_policies.hpp`.

4.247.2 Member Typedef Documentation

4.247.2.1 `template<typename Char, typename Char_Traits> typedef trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >::type`

Dispatched type.

Definition at line 149 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.248 `__gnu_pbds::detail::default_update_policy` Struct Reference

Public Types

- typedef [lu_move_to_front_policy](#) type

4.248.1 Detailed Description

Default update policy.

Definition at line 109 of file `standard_policies.hpp`.

4.248.2 Member Typedef Documentation

4.248.2.1 `typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type`

Dispatched type.

Definition at line 112 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.249 `__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >` Struct Template Reference

Public Types

- `typedef const_iterator` **const_reference**
- `typedef const_reference` **reference**
- `typedef const_iterator` **value_type**

4.249.1 Detailed Description

`template<typename Key, typename Data, typename _Alloc> struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >`

Constant node iterator.

Definition at line 52 of file `null_node_metadata.hpp`.

The documentation for this struct was generated from the following file:

- [null_node_metadata.hpp](#)

4.250 `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >` Struct Template Reference

Classes

- struct [type](#)
Compare plus entry.

Public Types

- `typedef`
`__rebind_v::other::const_pointer` **entry**

4.250.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>
```

Specialization, false.

Definition at line 62 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.251 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type` Struct Reference

Inherits `Cmp_Fn`.

Public Member Functions

- **type** (const `Cmp_Fn` &other)
- bool **operator()** (entry lhs, entry rhs) const

4.251.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type
```

Compare plus entry.

Definition at line 71 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.252 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>` Struct Template Reference

Public Types

- typedef `Cmp_Fn` [type](#)

4.252.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>
```

Specialization, true.

Definition at line 54 of file `entry_cmp.hpp`.

4.252.2 Member Typedef Documentation

4.252.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type`

Compare.

Definition at line 57 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.253 `__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >` Struct Template Reference

Public Types

- typedef
`__rebind_v::other::const_pointer` **entry**

4.253.1 Detailed Description

`template<typename _VTp, typename Pred, typename _Alloc> struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >`

Specialization, false.

Definition at line 61 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry_pred.hpp](#)

4.254 `__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >` Struct Template Reference

Public Types

- typedef `Pred` **type**

4.254.1 Detailed Description

`template<typename _VTp, typename Pred, typename _Alloc> struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >`

Specialization, true.

Definition at line 54 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry_pred.hpp](#)

4.255 `__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >` Struct Template Reference

Inherits `Cmp_Fn`.

Public Member Functions

- `bool operator()` (`const Key &r_lhs, const Key &r_rhs`) `const`

4.255.1 Detailed Description

`template<typename Key, class Cmp_Fn>struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >`

Equivalence function.

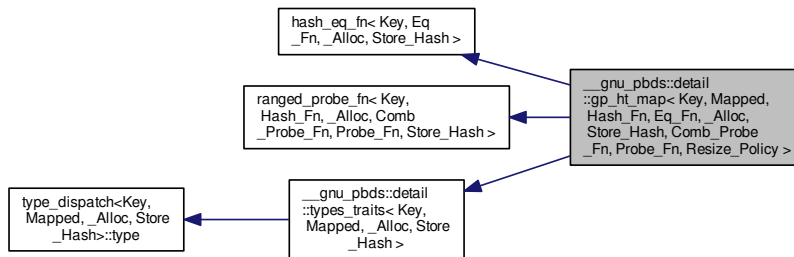
Definition at line 56 of file `eq_by_less.hpp`.

The documentation for this struct was generated from the following file:

- [eq_by_less.hpp](#)

4.256 `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`:



Public Types

- `enum { store_hash }`
- `typedef _Alloc allocator_type`
- `typedef Comb_Probe_Fn comb_probe_fn`
- `typedef const_iterator const_iterator`
- `typedef traits_base::const_pointer const_pointer`
- `typedef traits_base::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef Eq_Fn eq_fn`
- `typedef Hash_Fn hash_fn`
- `typedef iterator iterator`
- `typedef traits_base::key_const_pointer key_const_pointer`

- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef [point_const_iterator](#) **point_const_iterator**
- typedef [point_iterator](#) **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef Probe_Fn **probe_fn**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **gp_ht_map** (const [gp_ht_map](#)< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)
- **gp_ht_map** (const Hash_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &, const Resize_Policy &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It > void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred > size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Probe_Fn & [get_comb_probe_fn](#) ()

- `const Comb_Probe_Fn & get_comb_probe_fn () const`
- `Eq_Fn & get_eq_fn ()`
- `const Eq_Fn & get_eq_fn () const`
- `Hash_Fn & get_hash_fn ()`
- `const Hash_Fn & get_hash_fn () const`
- `Probe_Fn & get_probe_fn ()`
- `const Probe_Fn & get_probe_fn () const`
- `Resize_Policy & get_resize_policy ()`
- `const Resize_Policy & get_resize_policy () const`
- `std::pair< point_iterator, bool > insert (const_reference r_val)`
- `size_type max_size () const`
- `mapped_reference operator[] (key_const_reference r_key)`
- `size_type size () const`
- `void swap (gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)`

Public Attributes

- `no_throw_indicator m_no_throw_copies_indicator`
- `store_extra m_store_extra_indicator`

Friends

- `class const_iterator_`
- `class iterator_`

4.256.1 Detailed Description

`template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy> class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`

A general-probing hash-based container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to<Key></code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Probe_Fn</i>	Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> .
<i>Probe_Fn</i>	Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

Definition at line 142 of file `gp_ht_map.hpp`.

4.256.2 Member Enumeration Documentation

4.256.2.1 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` anonymous enum

Value stores hash, true or false.

Definition at line 208 of file `gp_ht_map.hpp`.

4.256.3 Member Function Documentation

4.256.3.1 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` `bool __gnu_pbds::detail::gp_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::empty () const`
[inline]

True if `size() == 0`.

Definition at line 58 of file `gp_ht_map.hpp`.

4.256.3.2 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` `Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::get_comb_probe_fn ()`

Return current `comb_probe_fn`.

Definition at line 82 of file `gp_ht_map.hpp`.

4.256.3.3 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` `const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::get_comb_probe_fn () const`

Return current `const comb_probe_fn`.

Definition at line 88 of file `gp_ht_map.hpp`.

4.256.3.4 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` `Eq_Fn & __gnu_pbds::detail::gp_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::get_eq_fn ()`

Return current `eq_fn`.

Definition at line 58 of file `gp_ht_map.hpp`.

4.256.3.5 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` `const Eq_Fn & __gnu_pbds::detail::gp_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::get_eq_fn () const`

Return current `const eq_fn`.

Definition at line 64 of file `gp_ht_map.hpp`.

4.256.3.6 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()`

Return current hash_fn.

Definition at line 46 of file `gp_ht_map.hpp`.

4.256.3.7 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const`

Return current const hash_fn.

Definition at line 52 of file `gp_ht_map.hpp`.

4.256.3.8 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()`

Return current probe_fn.

Definition at line 70 of file `gp_ht_map.hpp`.

4.256.3.9 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const`

Return current const probe_fn.

Definition at line 76 of file `gp_ht_map.hpp`.

4.256.3.10 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ()`

Return current resize_policy.

Definition at line 94 of file `gp_ht_map.hpp`.

4.256.3.11 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const`

Return current const resize_policy.

Definition at line 100 of file `gp_ht_map.hpp`.

The documentation for this class was generated from the following files:

- [gp_ht_map.hpp](#)
- [gp_hash_table_map_/constructor_destructor_fn_imps.hpp](#)
- [gp_hash_table_map_/find_fn_imps.hpp](#)

- [gp_hash_table_map_/resize_fn_imps.hpp](#)
- [gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp](#)
- [gp_hash_table_map_/resize_store_hash_fn_imps.hpp](#)
- [gp_hash_table_map_/info_fn_imps.hpp](#)
- [gp_hash_table_map_/policy_access_fn_imps.hpp](#)
- [gp_hash_table_map_/erase_fn_imps.hpp](#)
- [gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp](#)
- [gp_hash_table_map_/erase_store_hash_fn_imps.hpp](#)
- [iterator_fn_imps.hpp](#)
- [gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp](#)
- [gp_hash_table_map_/insert_store_hash_fn_imps.hpp](#)

4.257 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >` Struct Template Reference

Inherits `Eq_Fn`.

Public Types

- typedef `Eq_Fn` **eq_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**

Public Member Functions

- **hash_eq_fn** (`const Eq_Fn &r_eq_fn`)
- **operator()** (`key_const_reference r_lhs_key, key_const_reference r_rhs_key`) `const`
- **swap** (`const hash_eq_fn &other`)

4.257.1 Detailed Description

`template<typename Key, typename Eq_Fn, typename _Alloc> struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >`

Specialization 1 - The client requests that hash values not be stored.

Definition at line 58 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

4.258 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >` Struct Template Reference

Inherits `Eq_Fn`.

Public Types

- typedef `Eq_Fn` **eq_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**
- typedef `_Alloc::size_type` **size_type**

Public Member Functions

- **hash_eq_fn** (`const Eq_Fn &r_eq_fn`)
- **operator()** (`key_const_reference r_lhs_key, size_type lhs_hash, key_const_reference r_rhs_key, size_type rhs_hash`) `const`
- **swap** (`const hash_eq_fn &other`)

4.258.1 Detailed Description

`template<typename Key, class Eq_Fn, class _Alloc>struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >`

Specialization 2 - The client requests that hash values be stored.

Definition at line 81 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

4.259 `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >` Class Template Reference

Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- `size_type` **get_size** () `const`
- **set_size** (`size_type size`)
- **swap** ([hash_load_check_resize_trigger_size_base](#) &other)

4.259.1 Detailed Description

`template<typename Size_Type>class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >`

Specializations.

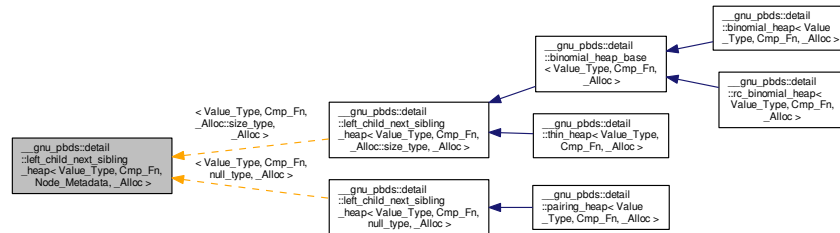
Definition at line 54 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash_load_check_resize_trigger_size_base.hpp](#)

4.260 `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >`:



Public Types

- `typedef _Alloc allocator_type`
- `typedef Cmp_Fn cmp_fn`
- `typedef`
`left_child_next_sibling_heap_const_iterator_`
`< node, _Alloc > const_iterator`
- `typedef`
`__rebind_v::other::const_pointer const_pointer`
- `typedef`
`__rebind_v::other::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef const_iterator iterator`
- `typedef`
`left_child_next_sibling_heap_node_point_const_iterator_`
`< node, _Alloc > point_const_iterator`
- `typedef point_const_iterator point_iterator`
- `typedef __rebind_v::other::pointer pointer`
- `typedef`
`__rebind_v::other::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef Value_Type value_type`

Public Member Functions

- `left_child_next_sibling_heap` (const Cmp_Fn &)
- `left_child_next_sibling_heap` (const `left_child_next_sibling_heap` &)
- `iterator begin` ()
- `const_iterator begin` () const
- void `clear` ()
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const

- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **max_size** () const
- size_type **size** () const
- void **swap** (`left_child_next_sibling_heap`< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &)

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template
rebind
< `left_child_next_sibling_heap_node_`
< Value_Type, Node_Metadata,
_Alloc > >::other **node_allocator**
- typedef
node_allocator::const_pointer **node_const_pointer**
- typedef Node_Metadata **node_metadata**
- typedef node_allocator::pointer **node_pointer**
- typedef `std::pair`
< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- node_pointer **get_new_node_for_insert** (const_reference)
- template<typename Pred >
node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (`left_child_next_sibling_heap` &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.260.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>class `__gnu_pbds::detail::left_child_next_sibling_heap`< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

Base class for a basic heap.

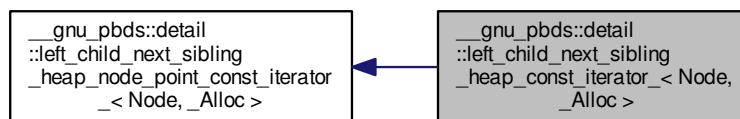
Definition at line 90 of file `left_child_next_sibling_heap_.hpp`.

The documentation for this class was generated from the following files:

- [left_child_next_sibling_heap_.hpp](#)
- [left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp](#)
- [left_child_next_sibling_heap_/iterators_fn_imps.hpp](#)
- [left_child_next_sibling_heap_/insert_fn_imps.hpp](#)
- [left_child_next_sibling_heap_/erase_fn_imps.hpp](#)
- [left_child_next_sibling_heap_/info_fn_imps.hpp](#)
- [left_child_next_sibling_heap_/policy_access_fn_imps.hpp](#)

4.261 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



Public Types

- typedef [base_type::const_pointer](#) `const_pointer`
- typedef [base_type::const_reference](#) `const_reference`
- typedef [_Alloc::difference_type](#) `difference_type`
- typedef [std::forward_iterator_tag](#) `iterator_category`
- typedef [base_type::pointer](#) `pointer`
- typedef [base_type::reference](#) `reference`
- typedef [base_type::value_type](#) `value_type`

Public Member Functions

- [left_child_next_sibling_heap_const_iterator_](#) (`node_pointer p_nd`)
- [left_child_next_sibling_heap_const_iterator_](#) ()
- [left_child_next_sibling_heap_const_iterator_](#) (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`)
- `bool operator!=` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`) `const`
- `bool operator!=` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`
- `const_reference operator*` () `const`
- [left_child_next_sibling_heap_const_iterator_](#) `< Node, _Alloc >` `& operator++` ()
- [left_child_next_sibling_heap_const_iterator_](#) `< Node, _Alloc >` `operator++` (`int`)

- `const_pointer operator-> () const`
- `bool operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const`
- `bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`

Public Attributes

- node_pointer `m_p_nd`

4.261.1 Detailed Description

`template<typename Node, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`

Const point-type iterator.

Definition at line 60 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2 Member Typedef Documentation

4.261.2.1 `template<typename Node , typename _Alloc > typedef base_type::const_pointer
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_pointer`

Iterator's const pointer type.

Definition at line 81 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2.2 `template<typename Node , typename _Alloc > typedef base_type::const_reference
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_reference`

Iterator's const reference type.

Definition at line 87 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2.3 `template<typename Node , typename _Alloc > typedef _Alloc::difference_type __gnu_pbds::detail::left_child_
next_sibling_heap_const_iterator_< Node, _Alloc >::difference_type`

Difference type.

Definition at line 72 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2.4 `template<typename Node , typename _Alloc > typedef std::forward_iterator_tag
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::iterator_category`

Category.

Definition at line 69 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2.5 `template<typename Node , typename _Alloc > typedef base_type::pointer __gnu_pbds::detail::left_child_
next_sibling_heap_const_iterator_< Node, _Alloc >::pointer`

Iterator's pointer type.

Definition at line 78 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2.6 `template<typename Node , typename _Alloc > typedef base_type::reference __gnu_pbds::detail::left_child_
next_sibling_heap_const_iterator_< Node, _Alloc >::reference`

Iterator's reference type.

Definition at line 84 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.2.7 `template<typename Node , typename _Alloc > typedef base_type::value_type
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::value_type`

Iterator's value type.

Definition at line 75 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.3 Constructor & Destructor Documentation

4.261.3.1 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling -
heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ ()
[inline]`

Default constructor.

Definition at line 96 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.3.2 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling -
heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ (const
left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) [inline]`

Copy constructor.

Definition at line 101 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.4 Member Function Documentation

4.261.4.1 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_const -
iterator< Node, _Alloc >::operator!=(const left_child_next_sibling_heap_const_iterator_< Node, _Alloc >
& other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 111 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.4.2 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child -
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!=(const
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) const [inline],
[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 137 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.261.4.3 `template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_child_next -
_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator*() const [inline],
[inherited]`

Access.

Definition at line 124 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.261.4.4 `template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> () const [inline], [inherited]`

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.261.4.5 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==(const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const [inline]`

Compares content to a different iterator object.

Definition at line 106 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.261.4.6 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator==(const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) const [inline], [inherited]`

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_/const_iterator.hpp](#)

4.262 `__gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct` Template Reference

Public Types

- typedef `_Metadata` **metadata_type**
- typedef `_Alloc::template rebind< this_type >::other::pointer` **node_pointer**
- typedef `_Alloc::size_type` **size_type**
- typedef `_Value` **value_type**

Public Attributes

- `metadata_type` **m_metadata**
- `node_pointer` **m_p_l_child**
- `node_pointer` **m_p_next_sibling**
- `node_pointer` **m_p_prev_or_parent**
- `value_type` **m_value**

4.262.1 Detailed Description

`template<typename _Value, typename _Metadata, typename _Alloc> struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >`

Node.

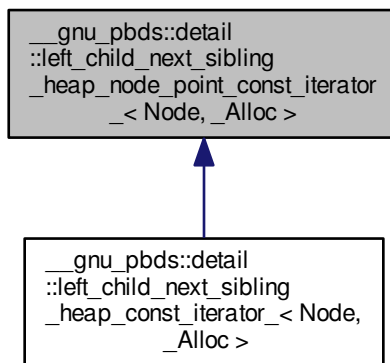
Definition at line 50 of file `left_child_next_sibling_heap_/node.hpp`.

The documentation for this struct was generated from the following file:

- [left_child_next_sibling_heap_/node.hpp](#)

4.263 `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`:



Public Types

- `typedef _Alloc::template rebind< value_type >::other::const_pointer const_pointer`
- `typedef _Alloc::template rebind< value_type >::other::const_reference const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef _Alloc::template rebind< value_type >::other::pointer pointer`
- `typedef _Alloc::template rebind< value_type >::other::reference reference`
- `typedef Node::value_type value_type`

Public Member Functions

- `left_child_next_sibling_heap_node_point_const_iterator_` (node_pointer p_nd)
- `left_child_next_sibling_heap_node_point_const_iterator_` ()
- `left_child_next_sibling_heap_node_point_const_iterator_` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other)
- `bool operator!=` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const
- `const_reference operator*` () const
- `const_pointer operator->` () const
- `bool operator==` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const

Public Attributes

- node_pointer `m_p_nd`

Protected Types

- `typedef _Alloc::template`
`rebind< Node >::other::pointer` `node_pointer`

4.263.1 Detailed Description

`template<typename Node, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`

Const point-type iterator.

Definition at line 61 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2 Member Typedef Documentation

4.263.2.1 `template<typename Node, typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_pointer`
`__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc`
`>::const_pointer`

Iterator's const pointer type.

Definition at line 86 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2.2 `template<typename Node, typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_reference`
`__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc`
`>::const_reference`

Iterator's const reference type.

Definition at line 98 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2.3 `template<typename Node, typename _Alloc > typedef trivial_iterator_difference_type`
`__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc`
`>::difference_type`

Difference type.

Definition at line 71 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2.4 `template<typename Node , typename _Alloc > typedef trivial_iterator_tag __gnu_pbds-
::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::iterator_category`

Category.

Definition at line 68 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2.5 `template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::pointer
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::pointer`

Iterator's pointer type.

Definition at line 80 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2.6 `template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::reference
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::reference`

Iterator's reference type.

Definition at line 92 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.2.7 `template<typename Node , typename _Alloc > typedef Node::value_type __gnu_pbds-
::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::value_type`

Iterator's value type.

Definition at line 74 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.3 Constructor & Destructor Documentation

4.263.3.1 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_
point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_()
[inline]`

Default constructor.

Definition at line 106 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.3.2 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_
point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_(
const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) [inline]`

Copy constructor.

Definition at line 111 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.263.4 Member Function Documentation

4.263.4.1 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!=(const
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 137 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.263.4.2  template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_
child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator* ( ) const
[inline]
```

Access.

Definition at line 124 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.263.4.3  template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child-
_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> ( ) const
[inline]
```

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.263.4.4  template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child -
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator== ( const
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_/point_const_iterator.hpp](#)

4.264 `__gnu_pbds::detail::lu_counter_metadata< Size_Type >` Class Template Reference

Public Types

- typedef `Size_Type` **size_type**

Friends

- class `lu_counter_policy_base< size_type >`

4.264.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

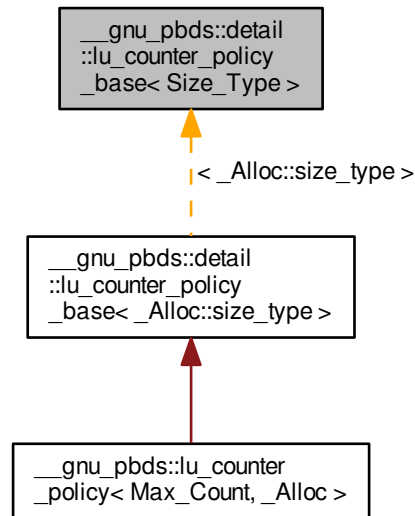
Definition at line 51 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu_counter_metadata.hpp](#)

4.265 `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >`:



Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- [lu_counter_metadata](#)< `size_type` > **operator()** (`size_type` max_size) const
- template<typename Metadata_Reference >
bool **operator()** (Metadata_Reference r_data, `size_type` m_max_count) const

4.265.1 Detailed Description

template<typename `Size_Type`>class `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >`

Base class for list-update counter policy.

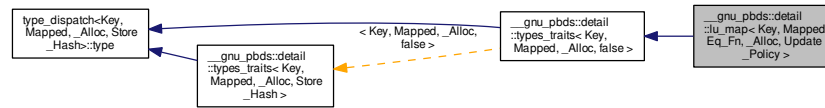
Definition at line 67 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu_counter_metadata.hpp](#)

4.266 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `const_iterator` **const_iterator**
- typedef `traits_base::const_pointer` **const_pointer**
- typedef `traits_base::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `Eq_Fn` **eq_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key_const_pointer**
- typedef `traits_base::key_const_reference` **key_const_reference**
- typedef `traits_base::key_pointer` **key_pointer**
- typedef `traits_base::key_reference` **key_reference**
- typedef `traits_base::key_type` **key_type**
- typedef `traits_base::mapped_const_pointer` **mapped_const_pointer**
- typedef `traits_base::mapped_const_reference` **mapped_const_reference**
- typedef `traits_base::mapped_pointer` **mapped_pointer**
- typedef `traits_base::mapped_reference` **mapped_reference**
- typedef `traits_base::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `point_const_iterator` **point_const_iterator**
- typedef `point_iterator` **point_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**
- typedef `Update_Policy::metadata_type` **update_metadata**
- typedef `Update_Policy` **update_policy**
- typedef `traits_base::value_type` **value_type**

Public Member Functions

- `lu_map` (const `lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` &)
- `template<typename It > lu_map` (It, It)
- iterator `begin` ()
- const_iterator `begin` () const
- void `clear` ()
- bool `empty` () const
- iterator `end` ()
- const_iterator `end` () const
- bool `erase` (key_const_reference)
- `template<typename Pred > size_type erase_if` (Pred)
- point_iterator `find` (key_const_reference r_key)
- point_const_iterator `find` (key_const_reference r_key) const
- `std::pair< point_iterator, bool > insert` (const_reference)
- size_type `max_size` () const
- mapped_reference `operator[]` (key_const_reference r_key)
- size_type `size` () const
- void `swap` (`lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` &)

Public Attributes

- no_throw_indicator `m_no_throw_copies_indicator`
- store_extra `m_store_extra_indicator`

Protected Member Functions

- `template<typename It > void copy_from_range` (It, It)

Friends

- class `const_iterator_`
- class `iterator_`

4.266.1 Detailed Description

`template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`

list-based (with updates) associative container. Skip to the lu, my darling.

Definition at line 91 of file `lu_map.hpp`.

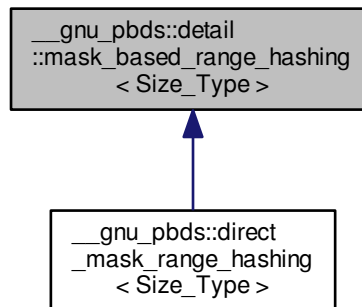
The documentation for this class was generated from the following files:

- `lu_map.hpp`
- `list_update_map_/constructor_destructor_fn_imps.hpp`
- `list_update_map_/info_fn_imps.hpp`

- [list_update_map_/iterators_fn_imps.hpp](#)
- [list_update_map_/erase_fn_imps.hpp](#)
- [list_update_map_/find_fn_imps.hpp](#)
- [list_update_map_/insert_fn_imps.hpp](#)

4.267 `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >`:



Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- void **notify_resized** (`size_type` size)
- `size_type` **range_hash** (`size_type` [hash](#)) const
- void **swap** ([mask_based_range_hashing](#) &other)

4.267.1 Detailed Description

`template<typename Size_Type> class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >`

Range hashing policy.

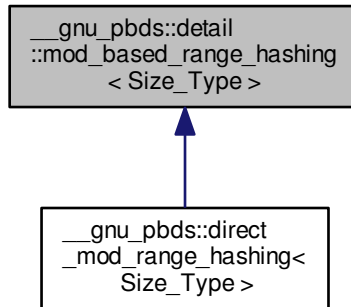
Definition at line 50 of file `mask_based_range_hashing.hpp`.

The documentation for this class was generated from the following file:

- [mask_based_range_hashing.hpp](#)

4.268 `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >`:



Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- void **notify_resized** (`size_type` s)
- `size_type` **range_hash** (`size_type` s) const
- void **swap** ([mod_based_range_hashing](#) &other)

4.268.1 Detailed Description

```
template<typename Size_Type>class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >
```

Mod based range hashing.

Definition at line 50 of file `mod_based_range_hashing.hpp`.

The documentation for this class was generated from the following file:

- [mod_based_range_hashing.hpp](#)

4.269 `__gnu_pbds::detail::no_throw_copies< Key, Mapped >` Struct Template Reference

Public Types

- typedef `integral_constant< int, __simple >` **indicator**

Static Public Attributes

- static const bool `__simple`

4.269.1 Detailed Description

```
template<typename Key, typename Mapped>struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
```

Primary template.

Definition at line 61 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.270 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference

Public Types

- typedef integral_constant< int, is_simple< Key >::value > **indicator**

4.270.1 Detailed Description

```
template<typename Key>struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
```

Specialization.

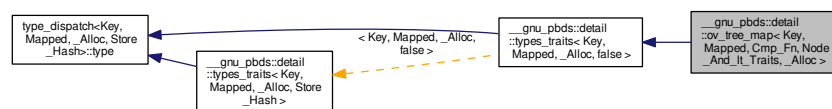
Definition at line 70 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.271 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`:



Classes

- class `cond_dtor`
Conditional destructor.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `point_const_iterator` **const_iterator**
- typedef `traits_base::const_pointer` **const_pointer**
- typedef `traits_base::const_reference` **const_reference**
- typedef `ov_tree_tag` **container_category**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key_const_pointer**
- typedef `traits_base::key_const_reference` **key_const_reference**
- typedef `traits_base::key_pointer` **key_pointer**
- typedef `traits_base::key_reference` **key_reference**
- typedef `traits_base::key_type` **key_type**
- typedef `traits_base::mapped_const_pointer` **mapped_const_pointer**
- typedef `traits_base::mapped_const_reference` **mapped_const_reference**
- typedef `traits_base::mapped_pointer` **mapped_pointer**
- typedef `traits_base::mapped_reference` **mapped_reference**
- typedef `traits_base::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `traits_type::node_const_iterator` **node_const_iterator**
- typedef `traits_type::node_iterator` **node_iterator**
- typedef `traits_type::node_update` **node_update**
- typedef `const_pointer` **point_const_iterator**
- typedef `pointer` **point_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**
- typedef `traits_base::value_type` **value_type**

Public Member Functions

- **ov_tree_map** (`const Cmp_Fn &`)
- **ov_tree_map** (`const Cmp_Fn &, const node_update &`)
- **ov_tree_map** (`const ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- iterator **begin** ()
- `const_iterator` **begin** () `const`
- void **clear** ()

- `template<typename It >`
`void copy_from_range (It, It)`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `bool erase (key_const_reference)`
- `iterator erase (iterator it)`
- `template<typename Pred >`
`size_type erase_if (Pred)`
- `point_iterator find (key_const_reference r_key)`
- `point_const_iterator find (key_const_reference r_key) const`
- `Cmp_Fn & get_cmp_fn ()`
- `const Cmp_Fn & get_cmp_fn () const`
- `std::pair< point_iterator, bool > insert (const_reference r_value)`
- `void join (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `point_iterator lower_bound (key_const_reference r_key)`
- `point_const_iterator lower_bound (key_const_reference r_key) const`
- `size_type max_size () const`
- `node_const_iterator node_begin () const`
- `node_iterator node_begin ()`
- `node_const_iterator node_end () const`
- `node_iterator node_end ()`
- `mapped_reference operator[] (key_const_reference r_key)`
- `size_type size () const`
- `void split (key_const_reference, ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `void swap (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `point_iterator upper_bound (key_const_reference r_key)`
- `point_const_iterator upper_bound (key_const_reference r_key) const`

Public Attributes

- no_throw_indicator `m_no_throw_copies_indicator`
- store_extra `m_store_extra_indicator`

4.271.1 Detailed Description

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`

Ordered-vector tree associative-container.

Definition at line 106 of file `ov_tree_map.hpp`.

4.271.2 Member Function Documentation

4.271.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator`
`__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const`
`[inline]`

Returns a const `node_iterator` corresponding to the node at the root of the tree.

Definition at line 45 of file `ov_tree_map.hpp`.

4.271.2.2 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits ,
typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ()
[inline]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 57 of file `ov_tree_map_.hpp`.

4.271.2.3 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const
[inline]`

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 51 of file `ov_tree_map_.hpp`.

4.271.2.4 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits ,
typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 63 of file `ov_tree_map_.hpp`.

The documentation for this class was generated from the following files:

- [ov_tree_map_.hpp](#)
- [ov_tree_map_/constructors_destructor_fn_imps.hpp](#)
- [ov_tree_map_/iterators_fn_imps.hpp](#)
- [ov_tree_map_/erase_fn_imps.hpp](#)
- [ov_tree_map_/insert_fn_imps.hpp](#)
- [ov_tree_map_/info_fn_imps.hpp](#)
- [ov_tree_map_/split_join_fn_imps.hpp](#)
- [bin_search_tree_/policy_access_fn_imps.hpp](#)

4.272 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

Public Member Functions

- `cond_dtor` (value_vector a_vec, iterator &r_last_it, Size_Type total_size)
- void `set_no_action` ()

Protected Attributes

- value_vector `m_a_vec`
- const Size_Type `m_max_size`
- bool `m_no_action`
- iterator & `m_r_last_it`

4.272.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>template<typename Size_Type>class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >
```

Conditional destructor.

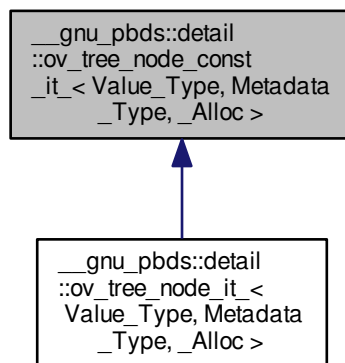
Definition at line 182 of file `ov_tree_map_.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map_.hpp](#)

4.273 `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >`:



Public Types

- `typedef _Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **const_reference**
- `typedef` [trivial_iterator_difference_type](#) **difference_type**
- `typedef` [trivial_iterator_tag](#) **iterator_category**
- `typedef _Alloc::template rebind< metadata_type >::other::const_reference` **metadata_const_reference**
- `typedef Metadata_Type` **metadata_type**

- `typedef _Alloc::template
rebind< typename remove_const
< Value_Type >::type >
::other::const_pointer` **reference**
- `typedef _Alloc::template
rebind< Value_Type >
::other::const_pointer` **value_type**

Public Member Functions

- `ov_tree_node_const_it_` (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- `this_type get_l_child ()` const
- `metadata_const_reference get_metadata ()` const
- `this_type get_r_child ()` const
- `bool operator!=` (const `this_type` &other) const
- `const_reference operator*` () const
- `bool operator==` (const `this_type` &other) const

Public Attributes

- pointer `m_p_begin_value`
- pointer `m_p_end_value`
- `const_metadata_pointer m_p_metadata`
- pointer `m_p_value`

Protected Types

- `typedef _Alloc::template
rebind< Metadata_Type >
::other::const_pointer` **const_metadata_pointer**
- `typedef _Alloc::template
rebind< Value_Type >
::other::const_pointer` **const_pointer**
- `typedef _Alloc::template
rebind< Value_Type >
::other::pointer` **pointer**
- `typedef ov_tree_node_const_it_
< Value_Type, Metadata_Type,
_Alloc >` **this_type**

Static Protected Member Functions

- `template<typename Ptr >
static Ptr` **mid_pointer** (Ptr p_begin, Ptr p_end)

4.273.1 Detailed Description

`template<typename Value_Type, typename Metadata_Type, typename _Alloc>class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >`

Const node reference.

Definition at line 57 of file `ov_tree_map_/node_iterators.hpp`.

4.273.2 Member Function Documentation

4.273.2.1 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_l_child () const
[inline]`

Returns the node iterator associated with the left node.

Definition at line 142 of file `ov_tree_map_/node_iterators.hpp`.

4.273.2.2 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_r_child () const
[inline]`

Returns the node iterator associated with the right node.

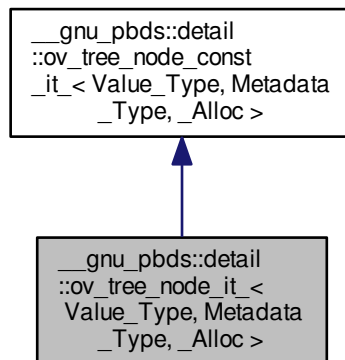
Definition at line 158 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map_/node_iterators.hpp](#)

4.274 `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`:



Public Types

- `typedef _Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer` **const_reference**
- `typedef trivial_iterator_difference_type` **difference_type**
- `typedef trivial_iterator_tag` **iterator_category**
- `typedef _Alloc::template rebind< metadata_type >::other::const_reference` **metadata_const_reference**
- `typedef Metadata_Type` **metadata_type**
- `typedef _Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer` **reference**
- `typedef _Alloc::template rebind< Value_Type >::other::pointer` **value_type**

Public Member Functions

- `ov_tree_node_it_` (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- `ov_tree_node_it_` get_l_child () const
- `metadata_const_reference` **get_metadata** () const
- `ov_tree_node_it_` get_r_child () const
- `bool` **operator!=** (const `this_type` &other) const
- `reference` **operator*** () const
- `bool` **operator==** (const `this_type` &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Static Protected Member Functions

- `template<typename Ptr >`
static `Ptr` **mid_pointer** (Ptr p_begin, Ptr p_end)

4.274.1 Detailed Description

`template<typename Value_Type, typename Metadata_Type, typename _Alloc>class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`

Node reference.

Definition at line 204 of file `ov_tree_map_/node_iterators.hpp`.

4.274.2 Member Function Documentation

4.274.2.1 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > ov_tree_node_it_
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get_l_child () const`
[inline]

Returns the node reference associated with the left node.

Definition at line 252 of file `ov_tree_map_/node_iterators.hpp`.

4.274.2.2 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > ov_tree_node_it_
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get_r_child () const`
[inline]

Returns the node reference associated with the right node.

Definition at line 268 of file `ov_tree_map_/node_iterators.hpp`.

4.274.2.3 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > reference
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* () const`
[inline]

Access.

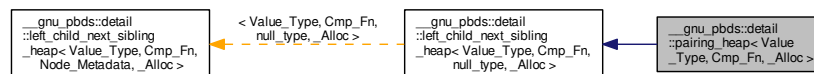
Definition at line 247 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map_/node_iterators.hpp](#)

4.275 `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_a::const_pointer` **const_pointer**
- typedef `__rebind_a::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_a::pointer` **pointer**

- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- **pairing_heap** (const Cmp_Fn &)
- **pairing_heap** (const [pairing_heap](#) &)
- **iterator begin** ()
- **const_iterator begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator end** ()
- **const_iterator end** () const
- void **erase** ([point_iterator](#))
- template<typename Pred >
size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** ([pairing_heap](#) &)
- size_type **max_size** () const
- void **modify** ([point_iterator](#), const_reference)
- void **pop** ()
- [point_iterator](#) **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, [pairing_heap](#) &)
- void **swap** ([pairing_heap](#) &)
- void **swap** ([left_child_next_sibling_heap](#)< Value_Type, Cmp_Fn, [null_type](#), _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef `node_allocator::value_type` **node**
- typedef `_Alloc::template
rebind
< left_child_next_sibling_heap_node
< Value_Type, null_type,
_Alloc > >::other` **node_allocator**
- typedef
`node_allocator::const_pointer` **node_const_pointer**
- typedef [null_type](#) **node_metadata**
- typedef [std::pair](#)
`< node_pointer, node_pointer >` **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.275.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`

Pairing heap.

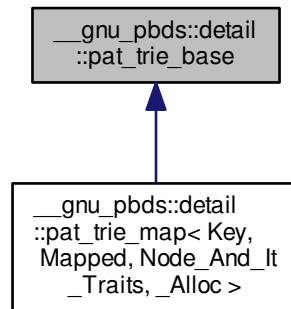
Definition at line 77 of file `pairing_heap_.hpp`.

The documentation for this class was generated from the following files:

- [pairing_heap_.hpp](#)
- [pairing_heap_/constructors_destructor_fn_imps.hpp](#)
- [pairing_heap_/find_fn_imps.hpp](#)
- [pairing_heap_/insert_fn_imps.hpp](#)
- [pairing_heap_/erase_fn_imps.hpp](#)
- [pairing_heap_/split_join_fn_imps.hpp](#)

4.276 __gnu_pbds::detail::pat_trie_base Struct Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base:



Classes

- class [_CIter](#)
Const iterator.
- struct [_Head](#)
Head node for PATRICIA tree.
- struct [_Inode](#)
Internal node type, PATRICIA tree.
- class [_Iter](#)
Iterator.
- struct [_Leaf](#)
Leaf node for PATRICIA tree.
- struct [_Metadata](#)
Metadata base primary template.
- struct [_Metadata< null_type, _Alloc >](#)
Specialization for null metadata.
- struct [_Node_base](#)
Node base.
- class [_Node_citer](#)
Node const iterator.
- class [_Node_iter](#)
Node iterator.

Public Types

- enum [node_type](#) { `i_node`, `leaf_node`, `head_node` }

4.276.1 Detailed Description

Base type for PATRICIA trees.

Definition at line 51 of file `pat_trie_base.hpp`.

4.276.2 Member Enumeration Documentation

4.276.2.1 `enum __gnu_pbds::detail::pat_trie_base::node_type`

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

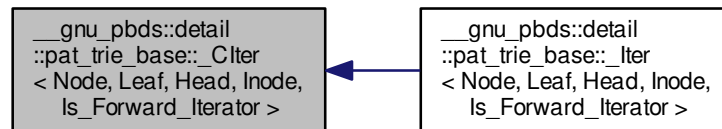
Definition at line 58 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.277 `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



Public Types

- `typedef _Alloc::template rebind< Head > __rebind_h`
- `typedef _Alloc::template rebind< Inode > __rebind_in`
- `typedef _Alloc::template rebind< Leaf > __rebind_l`
- `typedef _Alloc::template rebind< Node > __rebind_n`
- `typedef allocator_type _Alloc`
- `typedef Node::allocator_type allocator_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`

- typedef allocator_type::difference_type **difference_type**
- typedef __rebind_h::other::pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef __rebind_in::other::pointer **inode_pointer**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef __rebind_l::other::const_pointer **leaf_const_pointer**
- typedef __rebind_l::other::pointer **leaf_pointer**
- typedef __rebind_n::other::pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef Node::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Clter** (node_pointer p_nd=0)
- **_Clter** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator!=** (const [_Clter](#) &other) const
- bool **operator!=** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const
- const_reference **operator*** () const
- [_Clter](#) & **operator++** ()
- [_Clter](#) **operator++** (int)
- [_Clter](#) & **operator--** ()
- [_Clter](#) **operator--** (int)
- const_pointer **operator->** () const
- [_Clter](#) & **operator=** (const [_Clter](#) &other)
- [_Clter](#) & **operator=** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator==** (const [_Clter](#) &other) const
- bool **operator==** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

4.277.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator> class __gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

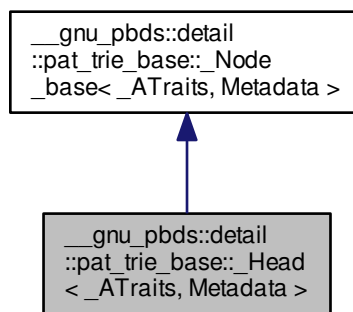
Definition at line 487 of file `pat_trie_base.hpp`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.278 `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>`:



Public Types

- typedef `_Alloc::template rebind<_ATraits>` **__rebind_at**
- typedef `_Alloc::template rebind<_Node_base>` **__rebind_n**
- typedef `_ATraits::const_iterator` **a_const_iterator**
- typedef `__rebind_at::other::const_pointer` **a_const_pointer**
- typedef `_ATraits` **access_traits**
- typedef `_Alloc` **allocator_type**
- typedef `_Node_base<_ATraits, Metadata>` **base_type**
- typedef `base_type::node_pointer` **node_pointer**
- typedef `base_type::type_traits` **type_traits**

Public Attributes

- node_pointer `m_p_max`
- node_pointer `m_p_min`
- node_pointer `m_p_parent`
- const [node_type](#) `m_type`

4.278.1 Detailed Description

`template<typename _ATraits, typename Metadata> struct __gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>`

Head node for PATRICIA tree.

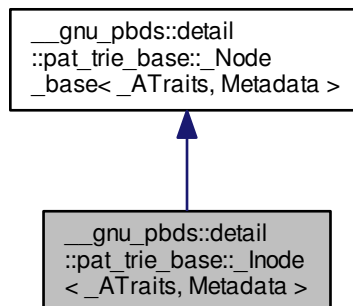
Definition at line 131 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.279 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>`:



Classes

- struct [const_iterator](#)
Constant child iterator.
- struct [iterator](#)
Child iterator.

Public Types

- enum { `arr_size` }

- `typedef _Alloc::template rebind<_ATraits> __rebind_at`
- `typedef _Alloc::template rebind<node_pointer>::other __rebind_np`
- `typedef base_type::allocator_type _Alloc`
- `typedef base_type::access_traits access_traits`
- `typedef _Alloc allocator_type`
- `typedef _Node_base<_ATraits, Metadata> base_type`
- `typedef __rebind_np::pointer node_pointer_pointer`
- `typedef __rebind_np::reference node_pointer_reference`
- `typedef _Alloc::size_type size_type`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

Public Member Functions

- `_Inode` (size_type, const a_const_iterator)
- `node_pointer add_child` (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- `const_iterator begin` () const
- `iterator begin` ()
- `const_iterator end` () const
- `iterator end` ()
- `iterator get_child_it` (a_const_iterator, a_const_iterator, a_const_pointer)
- `node_pointer get_child_node` (a_const_iterator, a_const_iterator, a_const_pointer)
- `node_const_pointer get_child_node` (a_const_iterator, a_const_iterator, a_const_pointer) const
- `size_type get_e_ind` () const
- `node_const_pointer get_join_child` (node_const_pointer, a_const_pointer) const
- `node_pointer get_join_child` (node_pointer, a_const_pointer)
- `node_pointer get_lower_bound_child_node` (a_const_iterator, a_const_iterator, size_type, a_const_pointer)
- `leaf_pointer leftmost_descendant` ()
- `leaf_const_pointer leftmost_descendant` () const
- `a_const_iterator pref_b_it` () const
- `a_const_iterator pref_e_it` () const
- `void remove_child` (node_pointer)
- `void remove_child` (iterator)
- `void replace_child` (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- `leaf_pointer rightmost_descendant` ()
- `leaf_const_pointer rightmost_descendant` () const
- `bool should_be_mine` (a_const_iterator, a_const_iterator, size_type, a_const_pointer) const
- `void update_prefixes` (a_const_pointer)

Public Attributes

- `node_pointer m_p_parent`
- `const node_type m_type`

4.279.1 Detailed Description

```
template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>
```

Internal node type, PATRICIA tree.

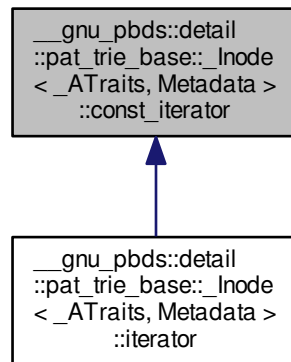
Definition at line 211 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.280 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator` Struct Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator`:



Public Types

- typedef `_Alloc::difference_type` **difference_type**
- typedef `std::forward_iterator_tag` **iterator_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value_type**

Public Member Functions

- **const_iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- **bool operator!=** (`const const_iterator &other`) `const`
- `node_const_pointer operator*` (`() const`)
- `const_iterator & operator++` (`()`)
- `const_iterator operator++` (`(int)`)
- `const node_pointer_pointer operator->` (`() const`)
- **bool operator==** (`const const_iterator &other`) `const`

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

4.280.1 Detailed Description

template<typename _ATraits, typename Metadata>struct `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator`

Constant child iterator.

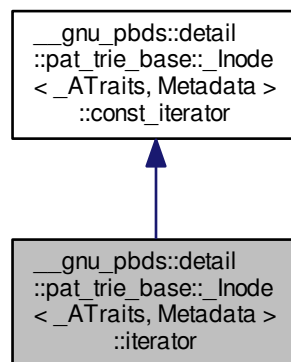
Definition at line 255 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.281 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator` Struct Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator`:



Public Types

- typedef `_Alloc::difference_type` **difference_type**
- typedef `std::forward_iterator_tag` **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- `iterator` (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- `bool operator!=` (const `const_iterator` &other) const
- `bool operator!=` (const `iterator` &other) const
- `node_const_pointer operator*` () const
- `node_pointer operator*` ()
- `iterator & operator++` ()
- `iterator operator++` (int)
- `const node_pointer_pointer operator->` () const
- `node_pointer_pointer operator->` ()
- `bool operator==` (const `const_iterator` &other) const
- `bool operator==` (const `iterator` &other) const

Public Attributes

- `node_pointer_pointer m_p_p_cur`
- `node_pointer_pointer m_p_p_end`

4.281.1 Detailed Description

`template<typename ATraits, typename Metadata> struct __gnu_pbds::detail::pat_trie_base::Inode< ATraits, Metadata >::iterator`

Child iterator.

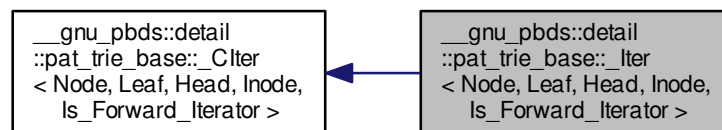
Definition at line 320 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.282 `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



Public Types

- `typedef _Alloc::template rebind< Head > __rebind_h`
- `typedef _Alloc::template rebind< Inode > __rebind_in`
- `typedef _Alloc::template rebind< Leaf > __rebind_l`
- `typedef _Alloc::template rebind< Node > __rebind_n`
- `typedef allocator_type _Alloc`
- `typedef base_type::allocator_type allocator_type`
- `typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef base_type::head_pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef base_type::inode_pointer inode_pointer`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef base_type::leaf_const_pointer leaf_const_pointer`
- `typedef base_type::leaf_pointer leaf_pointer`
- `typedef base_type::node_pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

Public Member Functions

- `_Iter (node_pointer p_nd=0)`
- `_Iter (const _Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other)`
- `bool operator!= (const _CIter &other) const`
- `bool operator!= (const _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other) const`
- `reference operator* () const`
- `_Iter & operator++ ()`
- `_Iter operator++ (int)`
- `_Iter & operator-- ()`
- `_Iter operator-- (int)`
- `pointer operator-> () const`
- `_Iter & operator= (const _Iter &other)`
- `_Iter & operator= (const _Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other)`
- `bool operator== (const _CIter &other) const`
- `bool operator== (const _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other) const`

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

4.282.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator> class `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`

Iterator.

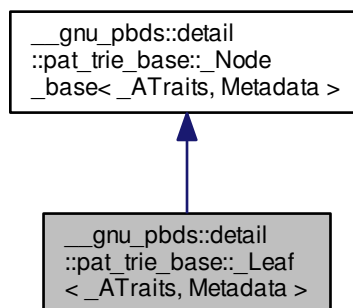
Definition at line 713 of file `pat_trie_base.hpp`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.283 `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>`:



Public Types

- typedef `_Alloc::template rebind< _ATraits > __rebind_at`
- typedef `_Alloc::template rebind< _Node_base > __rebind_n`
- typedef `_ATraits::const_iterator a_const_iterator`
- typedef `__rebind_at::other::const_pointer a_const_pointer`
- typedef `_ATraits access_traits`
- typedef `_Alloc allocator_type`
- typedef `_Node_base< _ATraits, Metadata > base_type`
- typedef `type_traits::const_reference const_reference`
- typedef `__rebind_n::other::pointer node_pointer`
- typedef `type_traits::reference reference`
- typedef `base_type::type_traits type_traits`
- typedef `type_traits::value_type value_type`

Public Member Functions

- `_Leaf` (const_reference other)
- reference `value` ()
- const_reference `value` () const

Public Attributes

- node_pointer `m_p_parent`
- const [node_type](#) `m_type`

4.283.1 Detailed Description

`template<typename _ATraits, typename Metadata> struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >`

Leaf node for PATRICIA tree.

Definition at line 162 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.284 `__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >` Struct Template Reference

Public Types

- typedef `_Alloc::template rebind< Metadata > __rebind_m`
- typedef `_Alloc allocator_type`
- typedef `__rebind_m::other::const_reference const_reference`
- typedef `Metadata metadata_type`

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

4.284.1 Detailed Description

`template<typename Metadata, typename _Alloc>struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >`

Metadata base primary template.

Definition at line 67 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.285 `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >` Struct Template Reference

Public Types

- typedef `_Alloc` **allocator_type**
- typedef [null_type](#) **metadata_type**

4.285.1 Detailed Description

`template<typename _Alloc>struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >`

Specialization for null metadata.

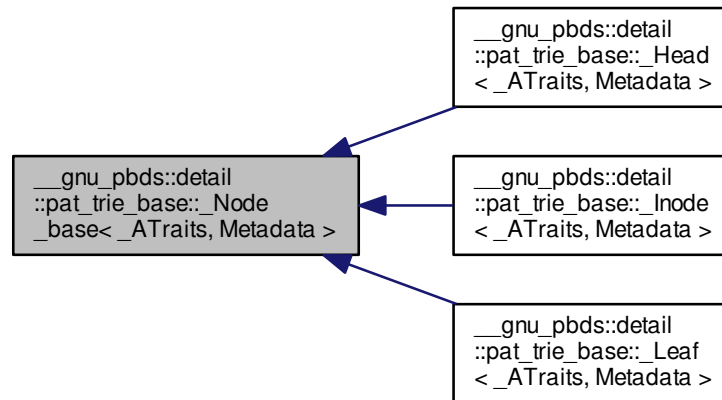
Definition at line 83 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.286 `__gnu_pbds::detail::pat_trie_base::_Node_base<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_base<_ATraits, Metadata>`:



Public Types

- typedef `_Alloc::template rebind<_ATraits>` **__rebind_at**
- typedef `_Alloc::template rebind<_Node_base>` **__rebind_n**
- typedef `_ATraits::const_iterator` **a_const_iterator**
- typedef `__rebind_at::other::const_pointer` **a_const_pointer**
- typedef `_ATraits` **access_traits**
- typedef `_Alloc` **allocator_type**
- typedef `__rebind_n::other::pointer` **node_pointer**
- typedef `_ATraits::type_traits` **type_traits**

Public Member Functions

- **`_Node_base`** ([node_type](#) type)

Public Attributes

- node_pointer **m_p_parent**
- const [node_type](#) **m_type**

4.286.1 Detailed Description

template<typename ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Node_base< ATraits, Metadata >

Node base.

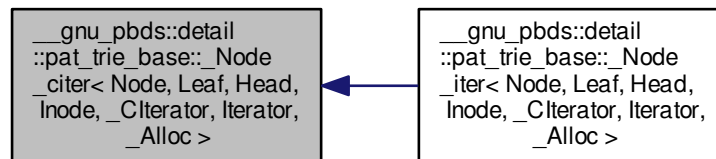
Definition at line 92 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.287 `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



Public Types

- typedef `_Alloc::template rebind< metadata_type > __rebind_m`
- typedef `__rebind_m::other __rebind_ma`
- typedef value_type **const_reference**
- typedef `trivial_iterator_difference_type difference_type`
- typedef `trivial_iterator_tag iterator_category`
- typedef `__rebind_ma::const_reference metadata_const_reference`
- typedef `Node::metadata_type metadata_type`
- typedef value_type **reference**
- typedef `_Alloc::size_type size_type`
- typedef `_CIterator value_type`

Public Member Functions

- **_Node_citer** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- [_Node_citer get_child](#) (size_type i) const

- metadata_const_reference [get_metadata](#) () const
- size_type [num_children](#) () const
- bool [operator!=](#) (const [_Node_citer](#) &other) const
- const_reference [operator*](#) () const
- bool [operator==](#) (const [_Node_citer](#) &other) const
- [std::pair](#)< a_const_iterator,
a_const_iterator > [valid_prefix](#) () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef `_Alloc::template rebind< Inode > __rebind_in`
- typedef `_Alloc::template rebind< Leaf > __rebind_l`
- typedef `_Alloc::template rebind< Node > __rebind_n`
- typedef `Node::a_const_iterator` **a_const_iterator**
- typedef `Node::a_const_pointer` **a_const_pointer**
- typedef
`__rebind_in::other::const_pointer` **inode_const_pointer**
- typedef `__rebind_in::other::pointer` **inode_pointer**
- typedef
`__rebind_l::other::const_pointer` **leaf_const_pointer**
- typedef `__rebind_l::other::pointer` **leaf_pointer**
- typedef `__rebind_n::other::pointer` **node_pointer**

4.287.1 Detailed Description

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _Citerator, typename Iterator, typename _Alloc>class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >`

Node const iterator.

Definition at line 814 of file `pat_trie_base.hpp`.

4.287.2 Member Typedef Documentation

4.287.2.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >::__rebind_m`

Const metadata reference type.

Definition at line 869 of file `pat_trie_base.hpp`.

4.287.2.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator ,
typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf,
Head, Inode, _Citerator, Iterator, _Alloc >::metadata_type`

Metadata type.

Definition at line 866 of file `pat_trie_base.hpp`.

4.287.3 Member Function Documentation

4.287.3.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator ,
typename _Alloc > _Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_Citerator, Iterator, _Alloc >::get_child (size_type i) const [inline]`

Returns a `__const` node `__iterator` to the corresponding node's `i`-th child.

Definition at line 911 of file `pat_trie_base.hpp`.

References `std::advance()`.

4.287.3.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator ,
typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf,
Head, Inode, _Citerator, Iterator, _Alloc >::get_metadata () const [inline]`

Metadata access.

Definition at line 894 of file `pat_trie_base.hpp`.

4.287.3.3 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator
, typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_Citerator, Iterator, _Alloc >::num_children () const [inline]`

Returns the number of children in the corresponding node.

Definition at line 899 of file `pat_trie_base.hpp`.

References `std::distance()`.

Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >::operator*()`, and `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >::operator*()`.

4.287.3.4 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator ,
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator,
Iterator, _Alloc >::operator!= (const _Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc > & other)
const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 927 of file `pat_trie_base.hpp`.

4.287.3.5 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _Citerator , typename Iterator ,
typename _Alloc > const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_Citerator, Iterator, _Alloc >::operator* () const [inline]`

Const access; returns the `__const` iterator* associated with the current leaf.

Definition at line 886 of file `pat_trie_base.hpp`.

References `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >-`

`::num_children()`.

4.287.3.6 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
 typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
 Iterator, _Alloc >::operator==(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
 const [inline]`

Compares content to a different iterator object.

Definition at line 922 of file `pat_trie_base.hpp`.

4.287.3.7 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename
 Iterator , typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat-
 _trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const
 [inline]`

Subtree valid prefix.

Definition at line 880 of file `pat_trie_base.hpp`.

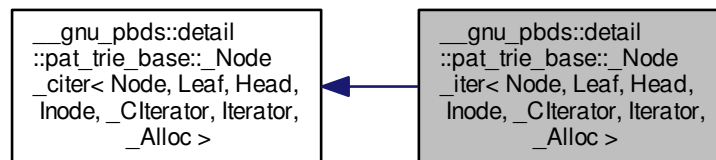
References `std::make_pair()`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.288 `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



Public Types

- `typedef _Alloc::template rebind< metadata_type > __rebind_m`
- `typedef __rebind_m::other __rebind_ma`
- `typedef value_type const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`

- typedef `__rebind_ma::const_reference` **metadata_const_reference**
- typedef `Node::metadata_type` [metadata_type](#)
- typedef `value_type` **reference**
- typedef `base_type::size_type` **size_type**
- typedef `Iterator` **value_type**

Public Member Functions

- **_Node_iter** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- [_Node_iter](#) [get_child](#) (size_type i) const
- `metadata_const_reference` [get_metadata](#) () const
- `size_type` [num_children](#) () const
- `bool` [operator!=](#) (const [_Node_citer](#) &other) const
- `reference` [operator*](#) () const
- `bool` [operator==](#) (const [_Node_citer](#) &other) const
- `std::pair< a_const_iterator, a_const_iterator >` [valid_prefix](#) () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef `_Alloc::template rebind< Inode >` **__rebind_in**
- typedef `_Alloc::template rebind< Leaf >` **__rebind_l**
- typedef `Node::a_const_iterator` **a_const_iterator**
- typedef `__rebind_in::other::const_pointer` **inode_const_pointer**
- typedef `__rebind_l::other::const_pointer` **leaf_const_pointer**
- typedef `__rebind_l::other::pointer` **leaf_pointer**

4.288.1 Detailed Description

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`

Node iterator.

Definition at line 943 of file `pat_trie_base.hpp`.

4.288.2 Member Typedef Documentation

4.288.2.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::_rebind_m [inherited]`

Const metadata reference type.

Definition at line 869 of file `pat_trie_base.hpp`.

4.288.2.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]`

Metadata type.

Definition at line 866 of file `pat_trie_base.hpp`.

4.288.3 Member Function Documentation

4.288.3.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > _Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child (size_type i) const [inline]`

Returns a node `__iterator` to the corresponding node's `i`-th child.

Definition at line 976 of file `pat_trie_base.hpp`.

References `std::advance()`, and `std::begin()`.

4.288.3.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata () const [inline],[inherited]`

Metadata access.

Definition at line 894 of file `pat_trie_base.hpp`.

4.288.3.3 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children () const [inline],[inherited]`

Returns the number of children in the corresponding node.

Definition at line 899 of file `pat_trie_base.hpp`.

References `std::distance()`.

Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*()`, and `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*()`.

4.288.3.4 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator!=(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other) const [inline],[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 927 of file `pat_trie_base.hpp`.

4.288.3.5 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::operator*() const [inline]`

Access; returns the iterator* associated with the current leaf.

Definition at line 968 of file `pat_trie_base.hpp`.

References `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children()`.

4.288.3.6 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator==(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]`

Compares content to a different iterator object.

Definition at line 922 of file `pat_trie_base.hpp`.

4.288.3.7 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator
, typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_
_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline],
[inherited]`

Subtree valid prefix.

Definition at line 880 of file `pat_trie_base.hpp`.

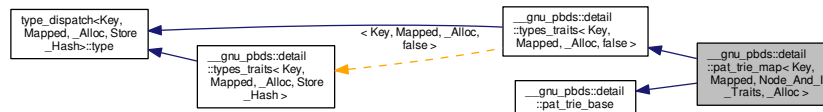
References `std::make_pair()`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.289 `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`:



Public Types

- typedef `traits_type::access_traits` **access_traits**
- typedef `_Alloc` **allocator_type**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `point_const_iterator` **const_iterator**

- `typedef traits_base::const_pointer` **const_pointer**
- `typedef traits_base::const_reference` **const_reference**
- `typedef traits_type::const_reverse_iterator` **const_reverse_iterator**
- `typedef pat_trie_tag container_category`
- `typedef _Alloc::difference_type` **difference_type**
- `typedef point_iterator` **iterator**
- `typedef traits_base::key_const_pointer` **key_const_pointer**
- `typedef traits_base::key_const_reference` **key_const_reference**
- `typedef traits_base::key_pointer` **key_pointer**
- `typedef traits_base::key_reference` **key_reference**
- `typedef traits_base::key_type` **key_type**
- `typedef traits_base::mapped_const_pointer` **mapped_const_pointer**
- `typedef traits_base::mapped_const_reference` **mapped_const_reference**
- `typedef traits_base::mapped_pointer` **mapped_pointer**
- `typedef traits_base::mapped_reference` **mapped_reference**
- `typedef traits_base::mapped_type` **mapped_type**
- `typedef __nothrowcopy::indicator` **no_throw_indicator**
- `typedef traits_type::node_const_iterator` **node_const_iterator**
- `typedef traits_type::node_iterator` **node_iterator**
- `enum node_type { i_node, leaf_node, head_node }`
- `typedef traits_type::node_update` **node_update**
- `typedef traits_type::const_iterator` **point_const_iterator**
- `typedef traits_type::iterator` **point_iterator**
- `typedef traits_base::pointer` **pointer**
- `typedef traits_base::reference` **reference**
- `typedef traits_type::reverse_iterator` **reverse_iterator**
- `typedef _Alloc::size_type` **size_type**
- `typedef integral_constant< int, Store_Hash >` **store_extra**
- `typedef traits_base::value_type` **value_type**

Public Member Functions

- **pat_trie_map** (const access_traits &)
- **pat_trie_map** (const [pat_trie_map](#)< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)

- `const_iterator` **erase** (`const_iterator`)
- `iterator` **erase** (`iterator`)
- `const_reverse_iterator` **erase** (`const_reverse_iterator`)
- `reverse_iterator` **erase** (`reverse_iterator`)
- `template<typename Pred >`
`size_type` **erase_if** (`Pred`)
- `point_iterator` **find** (`key_const_reference`)
- `point_const_iterator` **find** (`key_const_reference`) `const`
- `access_traits` & **get_access_traits** ()
- `const access_traits` & **get_access_traits** () `const`
- `node_update` & **get_node_update** ()
- `const node_update` & **get_node_update** () `const`
- `std::pair< point_iterator, bool >` **insert** (`const_reference`)
- `void` **join** (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)
- `point_iterator` **lower_bound** (`key_const_reference`)
- `point_const_iterator` **lower_bound** (`key_const_reference`) `const`
- `size_type` **max_size** () `const`
- `node_const_iterator` **node_begin** () `const`
- `node_iterator` **node_begin** ()
- `node_const_iterator` **node_end** () `const`
- `node_iterator` **node_end** ()
- `mapped_reference` **operator[]** (`key_const_reference r_key`)
- `reverse_iterator` **rbegin** ()
- `const_reverse_iterator` **rbegin** () `const`
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () `const`
- `size_type` **size** () `const`
- `void` **split** (`key_const_reference`, `pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)
- `void` **swap** (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)
- `point_iterator` **upper_bound** (`key_const_reference`)
- `point_const_iterator` **upper_bound** (`key_const_reference`) `const`

Public Attributes

- `no_throw_indicator` **m_no_throw_copies_indicator**
- `store_extra` **m_store_extra_indicator**

Protected Member Functions

- `template<typename It >`
`void` **copy_from_range** (`It`, `It`)
- `node_pointer` **recursive_copy_node** (`node_const_pointer`)
- `void` **value_swap** (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)

4.289.1 Detailed Description

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >
```

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Definition at line 101 of file `pat_trie_.hpp`.

4.289.2 Member Enumeration Documentation

4.289.2.1 `enum __gnu_pbds::detail::pat_trie_base::node_type` `[inherited]`

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

Definition at line 58 of file `pat_trie_base.hpp`.

4.289.3 Member Function Documentation

4.289.3.1 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const` `[inline]`

Returns a `const_node_iterator` corresponding to the node at the root of the tree.

Definition at line 101 of file `pat_trie_.hpp`.

4.289.3.2 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ()` `[inline]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 107 of file `pat_trie_.hpp`.

4.289.3.3 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const` `[inline]`

Returns a `const_node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 113 of file `pat_trie_.hpp`.

4.289.3.4 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ()` `[inline]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 119 of file `pat_trie_.hpp`.

The documentation for this class was generated from the following files:

- [pat_trie_.hpp](#)
- [pat_trie_/constructors_destructor_fn_imps.hpp](#)
- [pat_trie_/iterators_fn_imps.hpp](#)
- [insert_join_fn_imps.hpp](#)
- [pat_trie_/erase_fn_imps.hpp](#)
- [pat_trie_/find_fn_imps.hpp](#)
- [pat_trie_/info_fn_imps.hpp](#)
- [pat_trie_/policy_access_fn_imps.hpp](#)
- [split_fn_imps.hpp](#)
- [update_fn_imps.hpp](#)

4.290 `__gnu_pbds::detail::probe_fn_base<_Alloc>` Class Template Reference

4.290.1 Detailed Description

`template<typename _Alloc> class __gnu_pbds::detail::probe_fn_base<_Alloc>`

Probe functor base.

Definition at line 52 of file `probe_fn_base.hpp`.

The documentation for this class was generated from the following file:

- [probe_fn_base.hpp](#)

4.291 `__gnu_pbds::detail::ranged_hash_fn<Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false>` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

Protected Types

- `typedef Comb_Hash_Fn comb_hash_fn_base`
- `typedef Hash_Fn hash_fn_base`
- `typedef _Alloc::template
rebind<Key>::other key_allocator`
- `typedef
key_allocator::const_reference key_const_reference`
- `typedef _Alloc::size_type size_type`

Protected Member Functions

- `ranged_hash_fn (size_type)`
- `ranged_hash_fn (size_type, const Hash_Fn &)`
- `ranged_hash_fn (size_type, const Hash_Fn &, const Comb_Hash_Fn &)`
- `void notify_resized (size_type)`
- `size_type operator() (key_const_reference) const`
- `void swap (ranged_hash_fn<Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false> &)`

4.291.1 Detailed Description

`template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >`

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.292 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

Protected Types

- typedef `Comb_Hash_Fn` **comb_hash_fn_base**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `Hash_Fn` **hash_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- **ranged_hash_fn** (`size_type`)
- **ranged_hash_fn** (`size_type`, `const Hash_Fn &`)
- **ranged_hash_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Hash_Fn &`)
- void **notify_resized** (`size_type`)
- **comp_hash operator()** (`key_const_reference`) `const`
- **comp_hash operator()** (`key_const_reference`, `size_type`) `const`
- void **swap** ([ranged_hash_fn](#)< `Key`, `Hash_Fn`, `_Alloc`, `Comb_Hash_Fn`, `true` > &)

4.292.1 Detailed Description

`template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >`

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

Definition at line 153 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.293 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

Inherits `Comb_Hash_Fn`.

Protected Types

- typedef `Comb_Hash_Fn` **comb_hash_fn_base**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- **ranged_hash_fn** (`size_type`)
- **ranged_hash_fn** (`size_type`, const `Comb_Hash_Fn` &)
- **ranged_hash_fn** (`size_type`, const [null_type](#) &, const `Comb_Hash_Fn` &)
- void **swap** (`ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >` &)

4.293.1 Detailed Description

`template<typename Key, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >`

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

Definition at line 255 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.294 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Comb_Hash_Fn`.

Protected Types

- typedef `Comb_Hash_Fn` **comb_hash_fn_base**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- **ranged_hash_fn** (`size_type`)
- **ranged_hash_fn** (`size_type`, const `Comb_Hash_Fn` &)
- **ranged_hash_fn** (`size_type`, const [null_type](#) &, const `Comb_Hash_Fn` &)
- void **swap** (`ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` &)

4.294.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
```

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

Definition at line 312 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.295 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

Protected Types

- typedef `Comb_Probe_Fn` **comb_probe_fn_base**
- typedef `Hash_Fn` **hash_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**
- typedef `Probe_Fn` **probe_fn_base**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- **ranged_probe_fn** (`size_type`)
- **ranged_probe_fn** (`size_type`, `const Hash_Fn &`)
- **ranged_probe_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`)
- **ranged_probe_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`, `const Probe_Fn &`)
- void **notify_resized** (`size_type`)
- `size_type` **operator()** (`key_const_reference`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`, `size_type`) `const`
- void **swap** ([ranged_probe_fn](#)< `Key`, `Hash_Fn`, `_Alloc`, `Comb_Probe_Fn`, `Probe_Fn`, `false` > &)

4.295.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

Specialization 1 The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.296 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

Protected Types

- typedef `Comb_Probe_Fn` **comb_probe_fn_base**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `Hash_Fn` **hash_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**
- typedef `Probe_Fn` **probe_fn_base**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- **ranged_probe_fn** (`size_type`)
- **ranged_probe_fn** (`size_type`, `const Hash_Fn &`)
- **ranged_probe_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`)
- **ranged_probe_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`, `const Probe_Fn &`)
- void **notify_resized** (`size_type`)
- **comp_hash operator()** (`key_const_reference`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`, `size_type`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`) `const`
- void **swap** ([ranged_probe_fn](#) `< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` `&`)

4.296.1 Detailed Description

`template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >`

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 176 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.297 `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >` Class Template Reference

Inherits `Comb_Probe_Fn`.

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef _Alloc::template
rebind< Key >::other **key_allocator**
- typedef
key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type size)
- **ranged_probe_fn** (size_type, const Comb_Probe_Fn &r_comb_probe_fn)
- **ranged_probe_fn** (size_type, const [null_type](#) &, const Comb_Probe_Fn &r_comb_probe_fn, const [null_type](#) &)
- void **swap** ([ranged_probe_fn](#) &other)

4.297.1 Detailed Description

template<typename Key, typename _Alloc, typename Comb_Probe_Fn>class `__gnu_pbds::detail::ranged_probe_fn`< Key, [null_type](#),
_Alloc, Comb_Probe_Fn, [null_type](#), false >

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

Definition at line 296 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.298 `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class` **Template Reference**

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef [std::pair](#)< size_type,
size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef
base_type::const_reverse_iterator **const_reverse_iterator**
- typedef [rb_tree_tag](#) **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef
base_type::key_const_pointer **key_const_pointer**

- typedef
base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef
base_type::mapped_const_pointer **mapped_const_pointer**
- typedef
base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef
traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int,
Store_Hash > **store_extra**
- typedef base_type::value_type **value_type**

Public Member Functions

- **rb_tree_map** (const Cmp_Fn &)
- **rb_tree_map** (const Cmp_Fn &, const node_update &)
- **rb_tree_map** (const [rb_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- iterator **erase** (iterator)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- [std::pair](#)< point_iterator, bool > **insert** (const_reference)
- void **join** ([rb_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator **node_begin** () const
- node_iterator **node_begin** ()
- node_const_iterator **node_end** () const
- node_iterator **node_end** ()
- mapped_reference **operator[]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, [rb_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** ([rb_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template
rebind< typename
traits_type::node >::other **node_allocator**
- typedef
traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef [types_traits](#)< Key,
Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **apply_update** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **apply_update** (node_pointer, Node_Update_*)
- [std::pair](#)< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- [std::pair](#)< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

- `size_type recursive_count (node_pointer) const`
- `void rotate_left (node_pointer)`
- `void rotate_parent (node_pointer)`
- `void rotate_right (node_pointer)`
- `void split_finish (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `bool split_prep (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `void update_min_max_for_erased_node (node_pointer)`
- `void update_to_top (node_pointer, null_node_update_pointer)`
- `template<typename Node_Update_>`
`void update_to_top (node_pointer, Node_Update_*)`
- `void value_swap (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`

Static Protected Member Functions

- `static void clear_imp (node_pointer)`

Protected Attributes

- `node_pointer m_p_head`
- `size_type m_size`

Static Protected Attributes

- `static node_allocator s_node_allocator`

4.298.1 Detailed Description

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

Definition at line 84 of file `rb_tree_.hpp`.

4.298.2 Member Function Documentation

4.298.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const`
`[inline], [inherited]`

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 109 of file `bin_search_tree_.hpp`.

4.298.2.2 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator`
`__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ()`
`[inline], [inherited]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 117 of file `bin_search_tree_.hpp`.

4.298.2.3 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator`
`__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const`
`[inline], [inherited]`

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 125 of file `bin_search_tree_.hpp`.

4.298.2.4 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator`
`__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()`
`[inline], [inherited]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following files:

- [rb_tree_.hpp](#)
- [rb_tree_map_/constructors_destructor_fn_imps.hpp](#)
- [rb_tree_map_/insert_fn_imps.hpp](#)
- [rb_tree_map_/erase_fn_imps.hpp](#)
- [rb_tree_map_/split_join_fn_imps.hpp](#)
- [rb_tree_map_/info_fn_imps.hpp](#)

4.299 `__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

Public Types

- `typedef _Alloc::template rebind< metadata_type >::other::const_reference` **metadata_const_reference**
- `typedef _Alloc::template rebind< metadata_type >::other::reference` **metadata_reference**
- `typedef Metadata` **metadata_type**
- `typedef _Alloc::template rebind< rb_tree_node_< Value_Type, Metadata, _Alloc > >::other::pointer` **node_pointer**
- `typedef Value_Type` **value_type**

Public Member Functions

- `metadata_const_reference` **get_metadata** () const
- `metadata_reference` **get_metadata** ()
- `bool` **special** () const

Public Attributes

- `metadata_type` **m_metadata**
- `node_pointer` **m_p_left**
- `node_pointer` **m_p_parent**
- `node_pointer` **m_p_right**
- `bool` **m_red**
- `value_type` **m_value**

4.299.1 Detailed Description

`template<typename Value_Type, class Metadata, typename _Alloc>struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >`

Node for Red-Black trees.

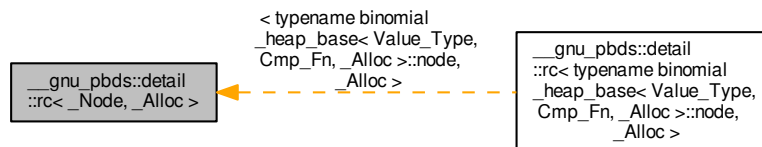
Definition at line 52 of file `rb_tree_map_/node.hpp`.

The documentation for this struct was generated from the following file:

- [rb_tree_map_/node.hpp](#)

4.300 `__gnu_pbds::detail::rc< _Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::rc< _Node, _Alloc >`:



Public Types

- `typedef entry_const_pointer` **const_iterator**
- `typedef node_pointer` **entry**

Public Member Functions

- `rc` (const `rc` &)
- `const_iterator` **begin** () const
- `void` **clear** ()
- `bool` **empty** () const
- `const_iterator` **end** () const
- `void` **pop** ()
- `void` **push** (entry)
- `size_type` **size** () const
- `void` **swap** (`rc` &)
- `node_pointer` **top** () const

4.300.1 Detailed Description

template<typename `_Node`, typename `_Alloc`>class `__gnu_pbds::detail::rc< _Node, _Alloc >`

Redundant binary counter.

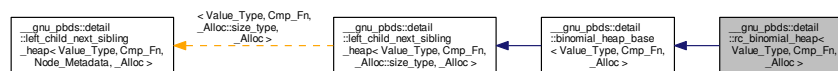
Definition at line 50 of file `rc.hpp`.

The documentation for this class was generated from the following file:

- `rc.hpp`

4.301 `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- `typedef base_type::allocator_type` **allocator_type**
- `typedef base_type::cmp_fn` **cmp_fn**
- `typedef base_type::const_iterator` **const_iterator**
- `typedef base_type::const_pointer` **const_pointer**
- `typedef base_type::const_reference` **const_reference**
- `typedef _Alloc::difference_type` **difference_type**
- `typedef base_type::iterator` **iterator**
- `typedef`
`base_type::point_const_iterator` **point_const_iterator**
- `typedef base_type::point_iterator` **point_iterator**
- `typedef base_type::pointer` **pointer**
- `typedef base_type::reference` **reference**
- `typedef _Alloc::size_type` **size_type**
- `typedef Value_Type` **value_type**

Public Member Functions

- **rc_binomial_heap** (const Cmp_Fn &)
- **rc_binomial_heap** (const [rc_binomial_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- void **erase** ([point_iterator](#))
- template<typename Pred >
size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** ([rc_binomial_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **join** ([binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** ([point_iterator](#), const_reference)
- void **pop** ()
- [point_iterator](#) **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, [rc_binomial_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- template<typename Pred >
void **split** (Pred, [binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** ([rc_binomial_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** ([left_child_next_sibling_heap](#)< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template
rebind
< [left_child_next_sibling_heap_node](#)
< Value_Type,
_Alloc::size_type, _Alloc >
>::other **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef [std::pair](#)
< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- void **find_max** ()

- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** ([binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.301.1 Detailed Description

`template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`

Redundant-counter binomial heap.

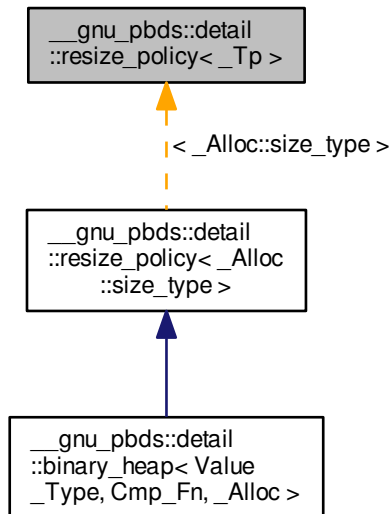
Definition at line 66 of file `rc_binomial_heap_.hpp`.

The documentation for this class was generated from the following files:

- [rc_binomial_heap_.hpp](#)
- [rc_binomial_heap_/constructors_destructor_fn_imps.hpp](#)
- [rc_binomial_heap_/erase_fn_imps.hpp](#)
- [rc_binomial_heap_/insert_fn_imps.hpp](#)
- [rc_binomial_heap_/split_join_fn_imps.hpp](#)

4.302 `__gnu_pbds::detail::resize_policy<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::resize_policy<_Tp>`:



Public Types

- typedef `_Tp` **size_type**

Public Member Functions

- **resize_policy** (const [resize_policy](#) &other)
- `size_type` **get_new_size_for_arbitrary** (`size_type`) const
- `size_type` **get_new_size_for_grow** () const
- `size_type` **get_new_size_for_shrink** () const
- `bool` **grow_needed** (`size_type`) const
- void **notify_arbitrary** (`size_type`)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- `bool` **resize_needed_for_grow** (`size_type`) const
- `bool` **resize_needed_for_shrink** (`size_type`) const
- `bool` **shrink_needed** (`size_type`) const
- void **swap** ([resize_policy](#)<_Tp> &)

Static Public Attributes

- static const `_Tp` **min_size**

4.302.1 Detailed Description

`template<typename _Tp>class __gnu_pbds::detail::resize_policy< _Tp >`

Resize policy for binary heap.

Definition at line 52 of file `resize_policy.hpp`.

The documentation for this class was generated from the following file:

- [resize_policy.hpp](#)

4.303 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `base_type::const_pointer` **const_pointer**
- typedef `base_type::const_reference` **const_reference**
- typedef `base_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `splay_tree_tag` **container_category**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key_const_pointer**
- typedef `base_type::key_const_reference` **key_const_reference**
- typedef `base_type::key_pointer` **key_pointer**
- typedef `base_type::key_reference` **key_reference**
- typedef `base_type::key_type` **key_type**
- typedef `base_type::mapped_const_pointer` **mapped_const_pointer**
- typedef `base_type::mapped_const_reference` **mapped_const_reference**
- typedef `base_type::mapped_pointer` **mapped_pointer**
- typedef `base_type::mapped_reference` **mapped_reference**
- typedef `base_type::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `traits_type::node_const_iterator` **node_const_iterator**
- typedef `traits_type::node_iterator` **node_iterator**
- typedef `base_type::node_update` **node_update**
- typedef `base_type::const_iterator` **point_const_iterator**

- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int,
Store_Hash > **store_extra**
- typedef base_type::value_type **value_type**

Public Member Functions

- **splay_tree_map** (const Cmp_Fn &)
- **splay_tree_map** (const Cmp_Fn &, const node_update &)
- **splay_tree_map** (const [splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- iterator **erase** (iterator it)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **initialize** ()
- [std::pair](#)< point_iterator, bool > **insert** (const_reference r_value)
- void **join** ([splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator **node_begin** () const
- node_iterator **node_begin** ()
- node_const_iterator **node_end** () const
- node_iterator **node_end** ()
- mapped_reference **operator[]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, [splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** ([splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template
rebind< typename
traits_type::node >::other **node_allocator**
- typedef
traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef [types_traits](#)< Key,
Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **apply_update** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **apply_update** (node_pointer, Node_Update_*)
- [std::pair](#)< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- [std::pair](#)< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **update_to_top** (node_pointer, Node_Update_*)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

4.303.1 Detailed Description

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`

Splay tree.

Definition at line 107 of file `splay_tree_.hpp`.

4.303.2 Member Function Documentation

4.303.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const`
`[inline], [inherited]`

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 109 of file `bin_search_tree_.hpp`.

4.303.2.2 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ()`
`[inline], [inherited]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 117 of file `bin_search_tree_.hpp`.

4.303.2.3 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const`
`[inline], [inherited]`

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 125 of file `bin_search_tree_.hpp`.

4.303.2.4 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()`
`[inline], [inherited]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following files:

- [splay_tree_.hpp](#)
- [splay_tree_/constructors_destructor_fn_imps.hpp](#)
- [splay_tree_/insert_fn_imps.hpp](#)
- [splay_fn_imps.hpp](#)
- [splay_tree_/erase_fn_imps.hpp](#)
- [splay_tree_/find_fn_imps.hpp](#)
- [splay_tree_/split_join_fn_imps.hpp](#)

4.304 `__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

Public Types

- `typedef _Alloc::template
rebind< metadata_type >
::other::const_reference metadata_const_reference`
- `typedef _Alloc::template
rebind< metadata_type >
::other::reference metadata_reference`
- `typedef Metadata metadata_type`
- `typedef _Alloc::template
rebind< splay_tree_node_
< Value_Type, Metadata, _Alloc >
>::other::pointer node_pointer`
- `typedef Value_Type value_type`

Public Member Functions

- `metadata_const_reference get_metadata () const`
- `metadata_reference get_metadata ()`
- `bool special () const`

Public Attributes

- `metadata_type m_metadata`
- `node_pointer m_p_left`
- `node_pointer m_p_parent`
- `node_pointer m_p_right`
- `bool m_special`
- `value_type m_value`

4.304.1 Detailed Description

`template<typename Value_Type, class Metadata, typename _Alloc>struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >`

Node for splay tree.

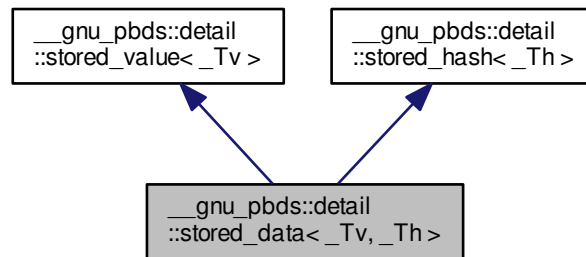
Definition at line 50 of file `splay_tree_/node.hpp`.

The documentation for this struct was generated from the following file:

- [splay_tree_/node.hpp](#)

4.305 `__gnu_pbds::detail::stored_data<_Tv, _Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, _Th>`:



Public Types

- typedef `_Th` **hash_type**
- typedef `_Tv` **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

4.305.1 Detailed Description

```
template<typename _Tv, typename _Th>struct __gnu_pbds::detail::stored_data<_Tv, _Th>
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.

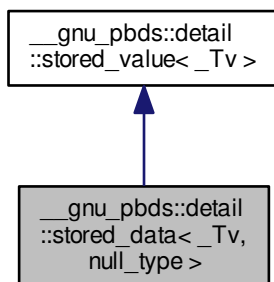
Definition at line 95 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.306 `__gnu_pbds::detail::stored_data<_Tv, null_type>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, null_type>`:

**Public Types**

- typedef `_Tv` **value_type**

Public Attributes

- value_type **m_value**

4.306.1 Detailed Description

```
template<typename _Tv>struct __gnu_pbds::detail::stored_data<_Tv, null_type>
```

Specialization for representation of stored data of just value type.

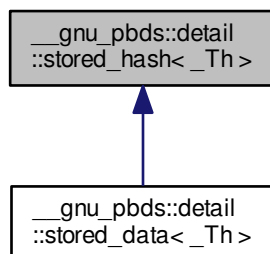
Definition at line 101 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.307 `__gnu_pbds::detail::stored_hash<_Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_hash<_Th>`:

**Public Types**

- `typedef _Th hash_type`

Public Attributes

- `hash_type m_hash`

4.307.1 Detailed Description

```
template<typename _Th>struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

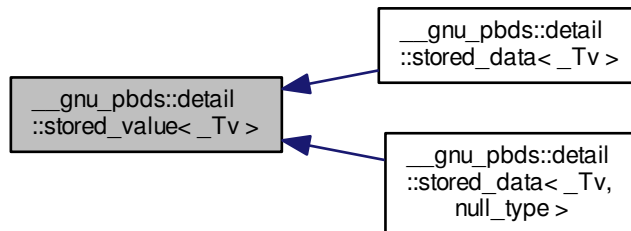
Definition at line 86 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.308 `__gnu_pbds::detail::stored_value<_Tv>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_value<_Tv>`:



Public Types

- typedef `_Tv` **value_type**

Public Attributes

- value_type **m_value**

4.308.1 Detailed Description

```
template<typename _Tv>struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

Definition at line 78 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.309 `__gnu_pbds::detail::synth_access_traits<Type_Traits, Set, _ATraits>` Struct Template Reference

Inherits `_ATraits`.

Public Types

- typedef `_ATraits` **base_type**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `type_traits::const_reference` **const_reference**
- typedef `type_traits::key_const_reference` **key_const_reference**

- typedef `Type_Traits` **type_traits**

Public Member Functions

- **synth_access_traits** (const `base_type` &)
- bool **cmp_keys** (key_const_reference, key_const_reference) const
- bool **cmp_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=false) const
- bool **equal_keys** (key_const_reference, key_const_reference) const
- bool **equal_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=true) const

Static Public Member Functions

- static key_const_reference **extract_key** (const_reference)

4.309.1 Detailed Description

template<typename `Type_Traits`, bool `Set`, typename `ATraits`>struct `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, ATraits >`

Synthetic element access traits.

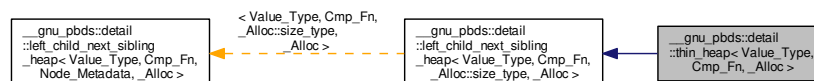
Definition at line 59 of file `synth_access_traits.hpp`.

The documentation for this struct was generated from the following file:

- [synth_access_traits.hpp](#)

4.310 `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_a::const_pointer` **const_pointer**
- typedef `__rebind_a::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**

- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred >
size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (`thin_heap< Value_Type, Cmp_Fn, _Alloc > &`)
- size_type **max_size** () const
- void **modify** (`point_iterator`, const_reference)
- void **pop** ()
- `point_iterator` **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, `thin_heap< Value_Type, Cmp_Fn, _Alloc > &`)
- void **swap** (`left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &`)
- const_reference **top** () const

Protected Types

- typedef `base_type::node` **node**
- typedef `_Alloc::template rebind
< left_child_next_sibling_heap_node_
< Value_Type,
_Alloc::size_type, _Alloc >
>::other` **node_allocator**
- typedef `base_type::node_const_pointer` **node_const_pointer**
- typedef `_Alloc::size_type` **node_metadata**
- typedef `base_type::node_pointer` **node_pointer**
- typedef `std::pair
< node_pointer, node_pointer >` **node_pointer_pair**

Protected Member Functions

- **thin_heap** (const Cmp_Fn &)
- **thin_heap** (const [thin_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** ([thin_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.310.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class `__gnu_pbds::detail::thin_heap`< Value_Type, Cmp_Fn, _Alloc >

Thin heap.

See Tarjan and Kaplan.

Definition at line 77 of file `thin_heap.hpp`.

The documentation for this class was generated from the following files:

- [thin_heap.hpp](#)
- [thin_heap_/constructors_destructor_fn_imps.hpp](#)
- [thin_heap_/find_fn_imps.hpp](#)
- [thin_heap_/insert_fn_imps.hpp](#)
- [thin_heap_/erase_fn_imps.hpp](#)
- [thin_heap_/split_join_fn_imps.hpp](#)

4.311 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

Public Types

- typedef Node_Update::metadata_type **type**

4.311.1 Detailed Description

```
template<typename Node_Update>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.312 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >` Struct Template Reference

Public Types

- typedef `null_type` **type**

4.312.1 Detailed Description

```
template<typename Node_Update>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.313 `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

Public Types

- typedef `tree_metadata_helper< __node_u, null_update >::type` **type**

4.313.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

Definition at line 84 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.314 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct` Template Reference

Public Types

- typedef [tree_node_metadata_dispatch](#)
`< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` **metadata_type**
- typedef [ov_tree_node_const_it_](#)
`< value_type, metadata_type, _Alloc >` **node_const_iterator**
- typedef [ov_tree_node_it_](#)
`< value_type, metadata_type, _Alloc >` **node_iterator**
- typedef `Node_Update`
`< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node_update**
- typedef [__gnu_pbds::null_node_update](#)
`< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > *` **null_node_update_pointer**

4.314.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Tree traits.

Definition at line 61 of file `ov_tree_map_/traits.hpp`.

4.314.2 Member Typedef Documentation

4.314.2.1 `template<typename Key , typename Mapped , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_ < value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

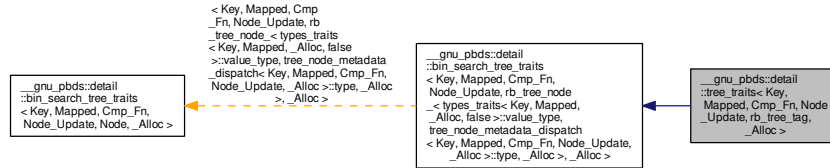
Definition at line 95 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov_tree_map_/traits.hpp](#)

4.315 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



Public Types

- typedef `bin_search_tree_const_iterator`
`< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > const_reverse_iterator`
- typedef `rb_tree_node`
`< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > node`
- typedef `bin_search_tree_const_node_iterator`
`< rb_tree_node < types_traits < Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator`
- typedef

```

    bin_search_tree_node_it_
    < rb_tree_node_< types_traits
    < Key, Mapped, _Alloc, false >
    ::value_type,
    tree_node_metadata_dispatch
    < Key, Mapped, Cmp_Fn,
    Node_Update, _Alloc >::type,
    _Alloc >, point_const_iterator,
    point_iterator, _Alloc > node_iterator
• typedef Node_Update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > node_update
• typedef
  __gnu_pbds::null_node_update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer
• typedef
  bin_search_tree_const_it_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_const_iterator
• typedef bin_search_tree_it_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_iterator
• typedef bin_search_tree_it_

```

```

< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > reverse_iterator

```

4.315.1 Detailed Description

```

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_-
Fn_, typename _Alloc_ > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_-
Update, rb_tree_tag, _Alloc >

```

Specialization.

Definition at line 61 of file `rb_tree_map_/traits.hpp`.

4.315.2 Member Typedef Documentation

4.315.2.1 `typedef bin_search_tree_const_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator` `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

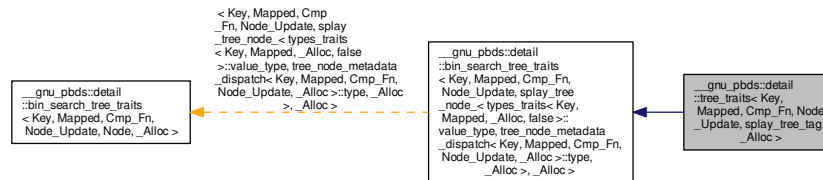
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb_tree_map_/traits.hpp](#)

4.316 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



Public Types

- typedef [bin_search_tree_const_it_](#)
`< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc >` **const_reverse_iterator**
- typedef [splay_tree_node_](#)
`< types_traits< Key, Mapped,
_Alloc, false >::value_type,
tree_node_metadata_dispatch
< Key, Mapped, Cmp_Fn,
Node_Update, _Alloc >::type,
_Alloc >` **node**
- typedef [bin_search_tree_const_node_it_](#)
`< splay_tree_node_
< types_traits< Key, Mapped,
_Alloc, false >::value_type,
tree_node_metadata_dispatch
< Key, Mapped, Cmp_Fn,
Node_Update, _Alloc >::type,
_Alloc >, point_const_iterator,
point_iterator, _Alloc >` **node_const_iterator**
- typedef

```

    bin_search_tree_node_it_
    < splay_tree_node_
    < types_traits< Key, Mapped,
    _Alloc, false >::value_type,
    tree_node_metadata_dispatch
    < Key, Mapped, Cmp_Fn,
    Node_Update, _Alloc >::type,
    _Alloc >, point_const_iterator,
    point_iterator, _Alloc > node_iterator
• typedef Node_Update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > node_update
• typedef
  __gnu_pbds::null_node_update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer
• typedef
  bin_search_tree_const_it_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_const_iterator
• typedef bin_search_tree_it_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_iterator
• typedef bin_search_tree_it_

```

```
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > reverse_iterator
```

4.316.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc_ > class Node_Update, typename _Alloc> struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`

Specialization.

Definition at line 61 of file `splay_tree_/traits.hpp`.

4.316.2 Member Typedef Documentation

4.316.2.1 `typedef bin_search_tree_const_node_it< splay_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator` `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [splay_tree_/traits.hpp](#)

4.317 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >` Struct Template Reference

Public Types

- typedef `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type` **metadata_type**
- typedef `ov_tree_node_const_it< value_type, metadata_type, _Alloc > node_const_iterator`
- typedef `node_const_iterator node_iterator`

- typedef Node_Update
`< node_const_iterator,
node_const_iterator, Cmp_Fn,
_Alloc > node_update`
- typedef
`__gnu_pbds::null_node_update
< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`

4.317.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`

Specialization.

Definition at line 132 of file `ov_tree_map_/traits.hpp`.

4.317.2 Member Typedef Documentation

4.317.2.1 `template<typename Key , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

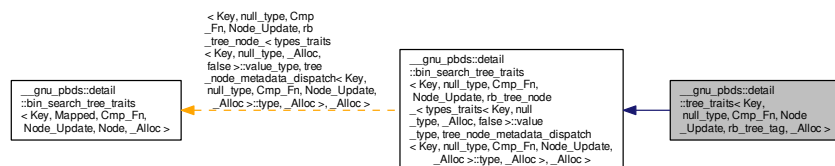
Definition at line 166 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov_tree_map_/traits.hpp](#)

4.318 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



Public Types

- typedef
[bin_search_tree_const_it_](#)
`< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > const_reverse_iterator`
- typedef [rb_tree_node_](#)
`< types_traits< Key, null_type,
_Alloc, false >::value_type,
tree_node_metadata_dispatch
< Key, null_type, Cmp_Fn,
Node_Update, _Alloc >::type,
_Alloc > node`
- typedef
[bin_search_tree_const_node_it_](#)
`< rb_tree_node_< types_traits
< Key, null_type, _Alloc,
false >::value_type,
tree_node_metadata_dispatch
< Key, null_type, Cmp_Fn,
Node_Update, _Alloc >::type,
_Alloc >, point_const_iterator,
point_iterator, _Alloc > node_const_iterator`
- typedef
[bin_search_tree_node_it_](#)
`< rb_tree_node_< types_traits
< Key, null_type, _Alloc,
false >::value_type,
tree_node_metadata_dispatch
< Key, null_type, Cmp_Fn,
Node_Update, _Alloc >::type,
_Alloc >, point_const_iterator,
point_iterator, _Alloc > node_iterator`
- typedef Node_Update
`< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > node_update`
- typedef
[__gnu_pbds::null_node_update](#)
`< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef

```

    bin_search_tree_const_it_
    < typename _Alloc::template
    rebind< node >::other::pointer,
    typename
    type_traits::value_type,
    typename type_traits::pointer,
    typename
    type_traits::const_pointer,
    typename
    type_traits::reference,
    typename
    type_traits::const_reference,
    true, _Alloc > point_const_iterator
• typedef bin\_search\_tree\_it\_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  true, _Alloc > point_iterator
• typedef bin\_search\_tree\_it\_
  < typename _Alloc::template
  rebind< node >::other::pointer,
  typename
  type_traits::value_type,
  typename type_traits::pointer,
  typename
  type_traits::const_pointer,
  typename
  type_traits::reference,
  typename
  type_traits::const_reference,
  false, _Alloc > reverse_iterator

```

4.318.1 Detailed Description

```

template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _
Alloc_ > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree-
_tag, _Alloc >

```

Specialization.

Definition at line 85 of file `rb_tree_map_/traits.hpp`.

4.318.2 Member Typedef Documentation

```

4.318.2.1 typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc
>, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key,
null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type,
tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc
>::node_const_iterator [inherited]

```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

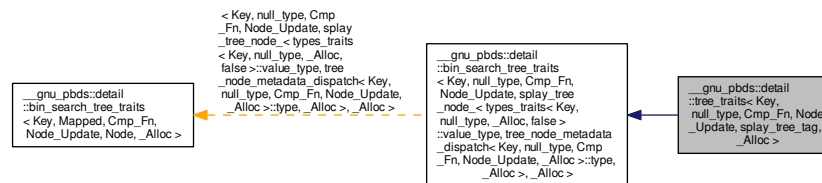
Definition at line 131 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- [rb_tree_map_/traits.hpp](#)

4.319 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef [bin_search_tree_const_it_](#)
< typename _Alloc::template
rebind< [node](#) >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > **const_reverse_iterator**
- typedef [splay_tree_node_](#)
< [types_traits](#)< Key, [null_type](#),
_Alloc, false >::value_type,
[tree_node_metadata_dispatch](#)
< Key, [null_type](#), Cmp_Fn,
Node_Update, _Alloc >::type,

-
- ```

 _Alloc > node
• typedef
 bin_search_tree_const_node_it_
 < splay_tree_node_
 < types_traits< Key, null_type,
 _Alloc, false >::value_type,
 tree_node_metadata_dispatch
 < Key, null_type, Cmp_Fn,
 Node_Update, _Alloc >::type,
 _Alloc >, point_const_iterator,
 point_iterator, _Alloc > node_const_iterator
• typedef
 bin_search_tree_node_it_
 < splay_tree_node_
 < types_traits< Key, null_type,
 _Alloc, false >::value_type,
 tree_node_metadata_dispatch
 < Key, null_type, Cmp_Fn,
 Node_Update, _Alloc >::type,
 _Alloc >, point_const_iterator,
 point_iterator, _Alloc > node_iterator
• typedef Node_Update
 < node_const_iterator,
 node_iterator, Cmp_Fn, _Alloc > node_update
• typedef
 __gnu_pbds::null_node_update
 < node_const_iterator,
 node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer
• typedef
 bin_search_tree_const_it_
 < typename _Alloc::template
 rebind< node >::other::pointer,
 typename
 type_traits::value_type,
 typename type_traits::pointer,
 typename
 type_traits::const_pointer,
 typename
 type_traits::reference,
 typename
 type_traits::const_reference,
 true, _Alloc > point_const_iterator
• typedef bin_search_tree_it_
 < typename _Alloc::template
 rebind< node >::other::pointer,
 typename
 type_traits::value_type,
 typename type_traits::pointer,
 typename
 type_traits::const_pointer,
 typename
 type_traits::reference,
 typename
 type_traits::const_reference,

```

```

true, _Alloc > point_iterator
• typedef bin_search_tree_it_
 < typename _Alloc::template
 rebind< node >::other::pointer,
 typename
 type_traits::value_type,
 typename type_traits::pointer,
 typename
 type_traits::const_pointer,
 typename
 type_traits::reference,
 typename
 type_traits::const_reference,
 false, _Alloc > reverse_iterator

```

#### 4.319.1 Detailed Description

```

template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn., typename _Alloc_ > class
Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc
>

```

Specialization.

Definition at line 81 of file `splay_tree_/traits.hpp`.

#### 4.319.2 Member Typedef Documentation

```

4.319.2.1 typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc
> , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key,
null_type , Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type,
tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , _Alloc
>::node_const_iterator [inherited]

```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

## 4.320 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >` Struct Template Reference

### Public Types

- typedef `Node_Update::metadata_type` **type**

#### 4.320.1 Detailed Description

```

template<typename Node_Update>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >

```

Specialization, false.

Definition at line 62 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.321 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >` Struct Template Reference

##### Public Types

- typedef `null_type` **type**

##### 4.321.1 Detailed Description

```
template<typename Node_Update>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
```

Specialization, `true`.

Definition at line 69 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.322 `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

##### Public Types

- typedef `trie_metadata_helper< __node_u, null_update >::type` **type**

##### 4.322.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Trie node metadata dispatch.

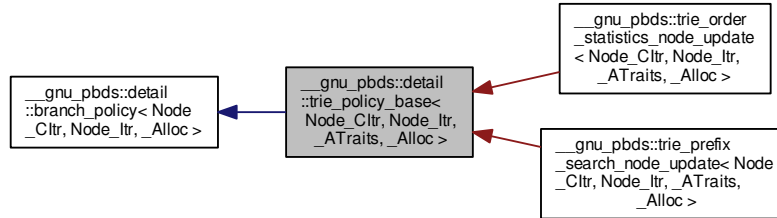
Definition at line 84 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

## 4.323 `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Protected Member Functions

- virtual `const_iterator` **end** () const =0
- virtual `iterator` **end** ()=0
- `it_type` **end\_iterator** () const
- virtual `const access_traits` & **get\_access\_traits** () const =0
- virtual `node_const_iterator` **node\_begin** () const =0
- virtual `node_iterator` **node\_begin** ()=0
- virtual `node_const_iterator` **node\_end** () const =0
- virtual `node_iterator` **node\_end** ()=0

## Static Protected Member Functions

- static `size_type` **common\_prefix\_len** (`node_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits` &)
- static `key_const_reference` **extract\_key** (`const_reference` r\_val)
- static `iterator` **leftmost\_it** (`node_iterator`)
- static `bool` **less** (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits` &)
- static `iterator` **rightmost\_it** (`node_iterator`)

## 4.323.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::detail::trie_policy_base<Node_Cltr, Node_Itr, _ATraits, _Alloc >`

Base class for trie policies.

Definition at line 53 of file `trie_policy_base.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy\\_base.hpp](#)

4.324 `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

## Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr` < `node`, `leaf`, `head`, `inode`, `true` > **const\_iterator**
- typedef `base_type::_Cltr` < `node`, `leaf`, `head`, `inode`, `false` > **const\_reverse\_iterator**
- typedef `base_type::_Head` < `synth_access_traits`, `metadata` > **head**
- typedef `base_type::_Inode` < `synth_access_traits`, `metadata` > **inode**
- typedef `base_type::_Itr` < `node`, `leaf`, `head`, `inode`, `true` > **iterator**

- typedef `base_type::_Leaf`  
  < `synth_access_traits`,  
  `metadata` > **leaf**
- typedef `base_type::_Metadata`  
  < `metadata_type`, `_Alloc` > **metadata**
- typedef  
  `trie_node_metadata_dispatch`  
  < `Key`, `Mapped`, `_ATraits`,  
  `Node_Update`, `_Alloc` >::type **metadata\_type**
- typedef `base_type::_Node_base`  
  < `synth_access_traits`,  
  `metadata` > **node**
- typedef `base_type::_Node_citer`  
  < `node`, `leaf`, `head`, `inode`,  
  `const_iterator`, `iterator`,  
  `_Alloc` > **node\_const\_iterator**
- typedef `base_type::_Node_iter`  
  < `node`, `leaf`, `head`, `inode`,  
  `const_iterator`, `iterator`,  
  `_Alloc` > **node\_iterator**
- typedef `Node_Update`  
  < `node_const_iterator`,  
  `node_iterator`, `_ATraits`,  
  `_Alloc` > **node\_update**
- typedef `null_node_update`  
  < `node_const_iterator`,  
  `node_iterator`, `_ATraits`,  
  `_Alloc` > \* **null\_node\_update\_pointer**
- typedef `base_type::_lter` < `node`,  
  `leaf`, `head`, `inode`, `false` > **reverse\_iterator**
- typedef  
  `__gnu_pbds::detail::synth_access_traits`  
  < `type_traits`, `false`,  
  `access_traits` > **synth\_access\_traits**

#### 4.324.1 Detailed Description

`template<typename Key, typename Mapped, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn., typename _Alloc. > class Node_Update, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`

Specialization.

Definition at line 62 of file `pat_trie_/traits.hpp`.

#### 4.324.2 Member Typedef Documentation

- 4.324.2.1 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn., typename _Alloc. > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 88 of file `pat_trie_/traits.hpp`.

```
4.324.2.2 template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr,
typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef
Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, Mapped,
_ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

Definition at line 93 of file `pat_trie_/traits.hpp`.

```
4.324.2.3 template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr,
typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef
__gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

Definition at line 74 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 4.325 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**

- typedef `base_type::_Node_citer`  
`< node, leaf, head, inode,`  
`const_iterator, iterator,`  
`_Alloc > node_const_iterator`
- typedef `node_const_iterator` `node_iterator`
- typedef `Node_Update`  
`< node_const_iterator,`  
`node_iterator, _ATraits,`  
`_Alloc > node_update`
- typedef `null_node_update`  
`< node_const_iterator,`  
`node_const_iterator, _ATraits,`  
`_Alloc > * null_node_update_pointer`
- typedef `const_reverse_iterator` `reverse_iterator`
- typedef  
`__gnu_pbds::detail::synth_access_traits`  
`< type_traits, true,`  
`access_traits > synth_access_traits`

#### 4.325.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_> class Node_Update, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 109 of file `pat_trie_/traits.hpp`.

#### 4.325.2 Member Typedef Documentation

4.325.2.1 `template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 135 of file `pat_trie_/traits.hpp`.

4.325.2.2 `template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update`

Type for node update.

Definition at line 140 of file `pat_trie_/traits.hpp`.

4.325.2.3 `template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits`

Type for synthesized traits.

Definition at line 121 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 4.326 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >` Struct Template Reference

### Public Types

- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef [stored\\_data](#)  
    `< value_type, null_type >` **stored\_data\_type**
- typedef `__rebind_va::value_type` **value\_type**

### 4.326.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc> struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`

Specialization of `type_base` for the case where the hash value is not stored alongside each value.

Definition at line 114 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.327 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >` Struct Template Reference

### Public Types

- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**

- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef [stored\\_data](#)  
`< value_type, size_type >` **stored\_data\_type**
- typedef `__rebind_va::value_type` **value\_type**

#### 4.327.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >`

Specialization of `type_base` for the case where the hash value is stored alongside each value.

Definition at line 147 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.328 `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >` Struct Template Reference

### Public Types

- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef [stored\\_data](#)  
`< value_type, null\_type >` **stored\_data\_type**
- typedef `Key` **value\_type**

### Static Public Attributes

- static [null\\_type](#) **s\_null\_type**

#### 4.328.1 Detailed Description

`template<typename Key, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >`

Specialization of `type_base` for the case where the hash value is not stored alongside each value.

Definition at line 181 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.329 `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >` Struct Template Reference

## Public Types

- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `__rebind_ma::value_type` **mapped\_type**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef [stored\\_data](#) `< value_type, size_type >` **stored\_data\_type**
- typedef `Key` **value\_type**

## Static Public Attributes

- static [null\\_type](#) **s\_null\_type**

## 4.329.1 Detailed Description

`template<typename Key, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`

Specialization of `type_base` for the case where the hash value is stored alongside each value.

Definition at line 220 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.330 `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

## Public Types

- typedef [type\\_base](#)`< Key, Mapped, _Alloc, Store_Hash >` **type**

## 4.330.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`

Type base dispatch.

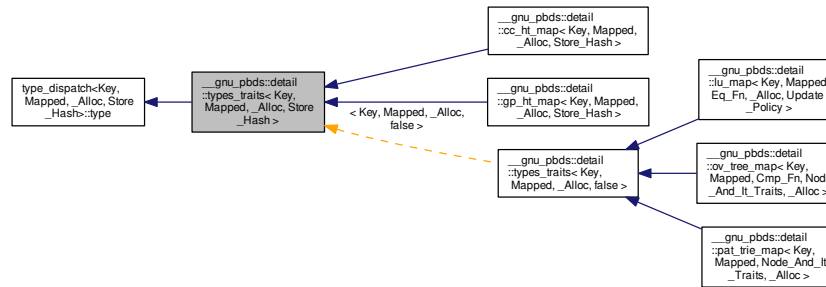
Definition at line 256 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.331 \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc, Store\_Hash > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc, Store\_Hash >:



### Public Types

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `__rebind_a::const_pointer` **key\_const\_pointer**
- typedef `__rebind_a::const_reference` **key\_const\_reference**
- typedef `__rebind_a::pointer` **key\_pointer**
- typedef `__rebind_a::reference` **key\_reference**
- typedef `__rebind_a::value_type` **key\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**

### Public Attributes

- `no_throw_indicator` **m\_no\_throw\_copies\_indicator**
- `store_extra` **m\_store\_extra\_indicator**

#### 4.331.1 Detailed Description

template<typename Key, typename Mapped, typename \_Alloc, bool Store\_Hash>struct \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc, Store\_Hash >

Traits for abstract types.

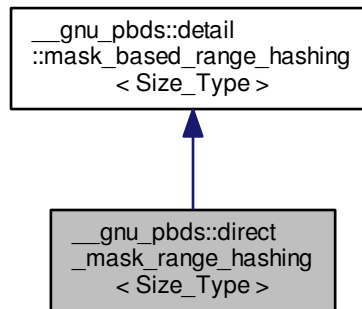
Definition at line 263 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.332 `__gnu_pbds::direct_mask_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mask_range_hashing< Size_Type >`:



## Public Types

- typedef `Size_Type` **size\_type**

## Public Member Functions

- void **swap** ([direct\\_mask\\_range\\_hashing< Size\\_Type >](#) &other)

## Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type [operator\(\)](#) (size\_type [hash](#)) const
- size\_type **range\_hash** (size\_type [hash](#)) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

## 4.332.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

Definition at line 109 of file `hash_policy.hpp`.

## 4.332.2 Member Function Documentation

4.332.2.1 `template<typename Size_Type > direct_mask_range_hashing< Size_Type >::size_type  
__gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() ( size_type hash ) const [inline],  
[protected]`

Transforms the \_\_hash value hash into a ranged-hash value (using a bit-mask).

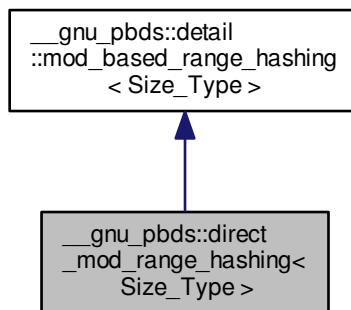
Definition at line 57 of file hash\_policy.hpp.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [direct\\_mask\\_range\\_hashing\\_imp.hpp](#)

### 4.333 \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size\_Type > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size\_Type >:



#### Public Types

- typedef Size\_Type **size\_type**

#### Public Member Functions

- void **swap** ([direct\\_mod\\_range\\_hashing](#)< Size\_Type > &other)

#### Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type [operator\(\)](#) (size\_type hash) const
- size\_type **range\_hash** (size\_type s) const
- void **swap** (mod\_based\_range\_hashing &other)

#### 4.333.1 Detailed Description

`template<typename Size_Type = std::size_t> class __gnu_pbds::direct_mod_range_hashing< Size_Type >`

A mod range-hashing class (uses the modulo function).

Definition at line 141 of file `hash_policy.hpp`.

#### 4.333.2 Member Function Documentation

4.333.2.1 `template<typename Size_Type > direct_mod_range_hashing< Size_Type >::size_type  
__gnu_pbds::direct_mod_range_hashing< Size_Type >::operator()( size_type hash ) const` `[inline]`,  
`[protected]`

Transforms the `__hash` value `hash` into a ranged-hash value (using a modulo operation).

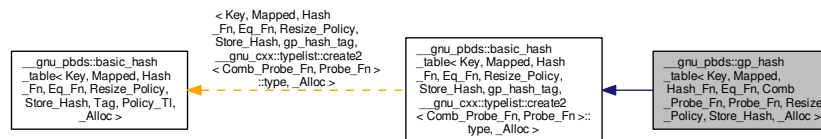
Definition at line 57 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [direct\\_mod\\_range\\_hashing\\_imp.hpp](#)

### 4.334 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- `typedef Comb_Probe_Fn comb_probe_fn`
- `typedef gp_hash_tag container_category`
- `typedef Eq_Fn eq_fn`
- `typedef Hash_Fn hash_fn`
- `typedef Probe_Fn probe_fn`
- `typedef Resize_Policy resize_policy`

#### Public Member Functions

- `gp_hash_table()`
- `gp_hash_table(const hash_fn &h)`

- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- template<typename It >  
`gp_hash_table` (It first, It last)
- template<typename It >  
`gp_hash_table` (It first, It last, const hash\_fn &h)
- template<typename It >  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It >  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- template<typename It >  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- template<typename It >  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- `gp_hash_table` (const `gp_hash_table` &other)
- `gp_hash_table` & **operator=** (const `gp_hash_table` &other)
- void **swap** (`gp_hash_table` &other)

## 4.334.1 Detailed Description

template<typename Key, typename Mapped, typename Hash\_Fn = typename detail::default\_hash\_fn<Key>::type, typename Eq\_Fn = typename detail::default\_eq\_fn<Key>::type, typename Comb\_Probe\_Fn = detail::default\_comb\_hash\_fn::type, typename Probe\_Fn = typename detail::default\_probe\_fn<Comb\_Probe\_Fn>::type, typename Resize\_Policy = typename detail::default\_resize\_policy<Comb\_Probe\_Fn>::type, bool Store\_Hash = detail::default\_store\_hash, typename \_Alloc = std::allocator<char>>> class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`

A general-probing hash-based associative container.

## Template Parameters

|                      |                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                               |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                               |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                        |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                          |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. |
| <i>Probe_Fn</i>      | Probe functor.                                                                                                                                                                                          |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                           |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If <i>Hash_Fn</i> is null_type, then the container will not compile if this value is true                                          |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                         |

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

Definition at line 368 of file `assoc_container.hpp`.

#### 4.334.2 Constructor & Destructor Documentation

4.334.2.1 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( ) [inline]`

Default constructor.

Definition at line 382 of file `assoc_container.hpp`.

4.334.2.2 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 386 of file `assoc_container.hpp`.

4.334.2.3 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h, const eq_fn & e ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 393 of file `assoc_container.hpp`.

4.334.2.4 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 401 of file `assoc_container.hpp`.

4.334.2.5 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 410 of file `assoc_container.hpp`.

4.334.2.6 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p, const resize_policy & rp ) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

Definition at line 422 of file `assoc_container.hpp`.

4.334.2.7 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last ) [inline]`

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 430 of file `assoc_container.hpp`.

4.334.2.8 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 438 of file `assoc_container.hpp`.

4.334.2.9 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 449 of file `assoc_container.hpp`.

4.334.2.10 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 461 of file `assoc_container.hpp`.

4.334.2.11 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 475 of file `assoc_container.hpp`.

4.334.2.12 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p, const resize_policy & rp ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container

object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

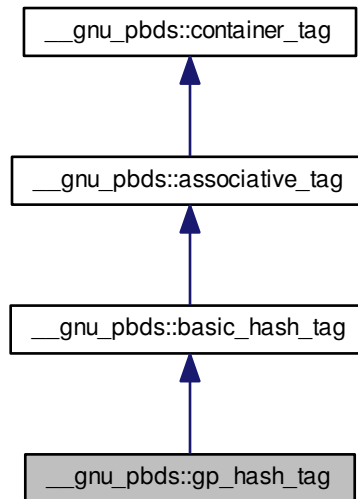
Definition at line 491 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 4.335 \_\_gnu\_pbds::gp\_hash\_tag Struct Reference

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



#### 4.335.1 Detailed Description

General-probing hash.

Definition at line 144 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.336 \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type > Class Template Reference

Public Types

- typedef `Size_Type` **size\_type**

#### Public Member Functions

- [hash\\_exponential\\_size\\_policy](#) (size\_type start\_size=8, size\_type grow\_factor=2)
- void **swap** ([hash\\_exponential\\_size\\_policy](#)< Size\_Type > &other)

#### Protected Member Functions

- size\_type **get\_nearest\_larger\_size** (size\_type size) const
- size\_type **get\_nearest\_smaller\_size** (size\_type size) const

#### 4.336.1 Detailed Description

`template<typename Size_Type = std::size_t>class __gnu_pbds::hash_exponential_size_policy< Size_Type >`

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).

Definition at line 413 of file `hash_policy.hpp`.

#### 4.336.2 Constructor & Destructor Documentation

4.336.2.1 `template<typename Size_Type > __gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy ( size_type start_size = 8, size_type grow_factor = 2 )`

Default constructor, or onstructor taking a start\_size, or constructor taking a start size and grow\_factor. The policy will use the sequence of sizes start\_size, start\_size\* grow\_factor, start\_size\* grow\_factor<sup>2</sup>, ...

Definition at line 44 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [hash\\_exponential\\_size\\_policy\\_imp.hpp](#)

#### 4.337 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

Inherits `hash_load_check_resize_trigger_size_base< Size_Type, External_Load_Access >`.

#### Public Types

- enum { [external\\_load\\_access](#) }
- typedef Size\_Type **size\_type**

#### Public Member Functions

- [hash\\_load\\_check\\_resize\\_trigger](#) (float load\_min=0.125, float load\_max=0.5)
- `std::pair< float, float >` [get\\_loads](#) () const
- void [set\\_loads](#) (`std::pair< float, float >` load\_pair)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger](#) &other)

#### Protected Member Functions

- bool **is\_grow\_needed** (size\_type size, size\_type num\_entries) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (size\_type num\_entries)
- void **notify\_externally\_resized** (size\_type new\_size)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** (size\_type num\_entries)
- void **notify\_resized** (size\_type new\_size)

#### 4.337.1 Detailed Description

`template<bool External_Load_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`

A resize trigger policy based on a load check. It keeps the load factor between some load factors `load_min` and `load_max`.

Definition at line 175 of file `hash_policy.hpp`.

#### 4.337.2 Member Enumeration Documentation

4.337.2.1 `template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum`

Enumerator:

***external\_load\_access*** Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 180 of file `hash_policy.hpp`.

#### 4.337.3 Constructor & Destructor Documentation

4.337.3.1 `template<bool External_Load_Access, typename Size_Type > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_resize_trigger ( float load_min = 0.125, float load_max = 0.5 )`

Default constructor, or constructor taking `load_min` and `load_max` load factors between which this policy will keep the actual load.

Definition at line 47 of file `hash_policy.hpp`.

## 4.337.4 Member Function Documentation

4.337.4.1 `template<bool External_Load_Access, typename Size_Type > std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::get_loads ( ) const`  
`[inline]`

Returns a pair of the minimal and maximal loads, respectively.

Definition at line 236 of file `hash_policy.hpp`.

4.337.4.2 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ( )`  
`[protected]`

Notifies the table was cleared.

Definition at line 206 of file `hash_policy.hpp`.

4.337.4.3 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted ( size_type num_entries )` `[inline]`,  
`[protected]`

Notifies an element was inserted. The total number of entries in the table is `num_entries`.

Definition at line 109 of file `hash_policy.hpp`.

4.337.4.4 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized ( size_type new_size )`  
`[protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 151 of file `hash_policy.hpp`.

4.337.4.5 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads ( std::pair< float, float > load_pair )`

Sets the loads through a pair of the minimal and maximal loads, respectively.

Definition at line 245 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [hash\\_load\\_check\\_resize\\_trigger\\_imp.hpp](#)

4.338 `__gnu_pbds::hash_prime_size_policy` Class Reference

## Public Types

- `typedef std::size_t` [size\\_type](#)

## Public Member Functions

- [hash\\_prime\\_size\\_policy](#) ([size\\_type](#) start\_size=8)
- `void` [swap](#) ([hash\\_prime\\_size\\_policy](#) &other)

#### Protected Member Functions

- [size\\_type](#) `get_nearest_larger_size` ([size\\_type](#) size) const
- [size\\_type](#) `get_nearest_smaller_size` ([size\\_type](#) size) const

#### 4.338.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

Definition at line 450 of file `hash_policy.hpp`.

#### 4.338.2 Member Typedef Documentation

##### 4.338.2.1 `typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type`

Size type.

Definition at line 454 of file `hash_policy.hpp`.

#### 4.338.3 Constructor & Destructor Documentation

##### 4.338.3.1 `__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy ( size_type start_size = 8 ) [inline]`

Default constructor, or onstructor taking a `start_size` The policy will use the sequence of sizes approximately `start_size`, `start_size* 2`, `start_size* 2^2`, ...

Definition at line 127 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [hash\\_prime\\_size\\_policy\\_imp.hpp](#)

### 4.339 `__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >` Class Template Reference

Inherits `Size_Policy`, and `Trigger_Policy`.

#### Public Types

- enum { **external\_size\_access** }
- typedef `Size_Policy` **size\_policy**
- typedef `Size_Type` **size\_type**
- typedef `Trigger_Policy` **trigger\_policy**

#### Public Member Functions

- [hash\\_standard\\_resize\\_policy](#) ()
- [hash\\_standard\\_resize\\_policy](#) (const `Size_Policy` &r\_size\_policy)
- [hash\\_standard\\_resize\\_policy](#) (const `Size_Policy` &r\_size\_policy, const `Trigger_Policy` &r\_trigger\_policy)
- `size_type` [get\\_actual\\_size](#) () const

- `Size_Policy` & [get\\_size\\_policy](#) ()
- const `Size_Policy` & [get\\_size\\_policy](#) () const
- `Trigger_Policy` & [get\\_trigger\\_policy](#) ()
- const `Trigger_Policy` & [get\\_trigger\\_policy](#) () const
- void [resize](#) (size\_type suggested\_new\_size)
- void **swap** ([hash\\_standard\\_resize\\_policy](#)< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` > &other)

#### Protected Member Functions

- size\_type [get\\_new\\_size](#) (size\_type size, size\_type num\_used\_e) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (size\_type num\_e)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** (size\_type num\_e)
- void **notify\_resized** (size\_type new\_size)

#### 4.339.1 Detailed Description

template<typename `Size_Policy` = `hash_exponential_size_policy`<>, typename `Trigger_Policy` = `hash_load_check_resize_trigger`<>, bool `External_Size_Access` = false, typename `Size_Type` = std::size\_t>class `__gnu_pbds::hash_standard_resize_policy`< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` >

A resize policy which delegates operations to size and trigger policies.

Definition at line 489 of file `hash_policy.hpp`.

#### 4.339.2 Constructor & Destructor Documentation

4.339.2.1 template<typename `Size_Policy` , typename `Trigger_Policy` , bool `External_Size_Access`, typename `Size_Type` >  
`__gnu_pbds::hash_standard_resize_policy`< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` >::`hash_standard_resize_policy` ( )

Default constructor.

Definition at line 44 of file `hash_policy.hpp`.

4.339.2.2 template<typename `Size_Policy`, typename `Trigger_Policy` , bool `External_Size_Access`, typename `Size_Type` >  
`__gnu_pbds::hash_standard_resize_policy`< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` >::`hash_standard_resize_policy` ( const `Size_Policy` & *r\_size\_policy* )

constructor taking some policies `r_size_policy` will be copied by the `Size_Policy` object of this object.

Definition at line 50 of file `hash_policy.hpp`.

4.339.2.3 `template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_Type > __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::hash_standard_resize_policy ( const Size_Policy & r_size_policy, const Trigger_Policy & r_trigger_policy )`

constructor taking some policies. `r_size_policy` will be copied by the `Size_Policy` object of this object. `r_trigger_policy` will be copied by the `Trigger_Policy` object of this object.

Definition at line 56 of file `hash_policy.hpp`.

#### 4.339.3 Member Function Documentation

4.339.3.1 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_actual_size ( ) const [inline]`

Returns the actual size of the container.

Definition at line 177 of file `hash_policy.hpp`.

4.339.3.2 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_new_size ( size_type size, size_type num_used_e ) const [protected]`

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

Definition at line 158 of file `hash_policy.hpp`.

4.339.3.3 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_size_policy ( )`

Access to the `Size_Policy` object used.

Definition at line 242 of file `hash_policy.hpp`.

4.339.3.4 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_size_policy ( ) const`

Const access to the `Size_Policy` object used.

Definition at line 248 of file `hash_policy.hpp`.

4.339.3.5 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy ( )`

Access to the `Trigger_Policy` object used.

Definition at line 230 of file `hash_policy.hpp`.

4.339.3.6 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy ( ) const`

Access to the `Trigger_Policy` object used.

Definition at line 236 of file `hash_policy.hpp`.

4.339.3.7 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::resize ( size_type suggested_new_size )`

Resizes the container to `suggested_new_size`, a suggested size (the actual size will be determined by the `Size_Policy` object).

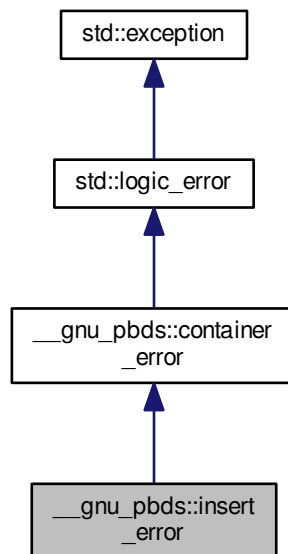
Definition at line 186 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [hash\\_standard\\_resize\\_policy\\_imp.hpp](#)

## 4.340 `__gnu_pbds::insert_error` Struct Reference

Inheritance diagram for `__gnu_pbds::insert_error`:



### Public Member Functions

- virtual const char \* `what` () const noexcept

## 4.340.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

Definition at line 66 of file `exception.hpp`.

## 4.340.2 Member Function Documentation

4.340.2.1 `virtual const char* std::logic_error::what ( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

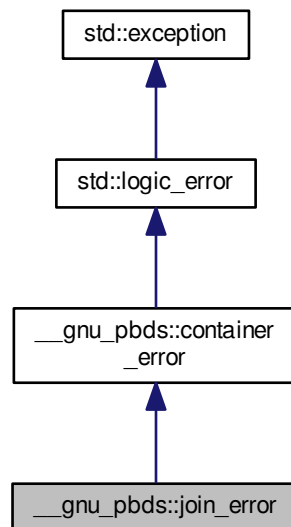
Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

## 4.341 \_\_gnu\_pbds::join\_error Struct Reference

Inheritance diagram for `__gnu_pbds::join_error`:



## Public Member Functions

- `virtual const char * what ( ) const noexcept`

## 4.341.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

Definition at line 70 of file `exception.hpp`.

## 4.341.2 Member Function Documentation

4.341.2.1 `virtual const char* std::logic_error::what( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

Reimplemented in `std::future_error`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.342 `__gnu_pbds::linear_probe_fn< Size_Type >` Class Template Reference

## Public Types

- typedef `Size_Type` **size\_type**

## Public Member Functions

- void **swap** ([linear\\_probe\\_fn< Size\\_Type >](#) &other)

## Protected Member Functions

- `size_type` [operator\(\)](#) (`size_type` i) const

## 4.342.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

Definition at line 61 of file `hash_policy.hpp`.

## 4.342.2 Member Function Documentation

4.342.2.1 `template<typename Size_Type > linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() ( size_type i ) const` `[inline]`, `[protected]`

Returns the i-th offset from the hash value.

Definition at line 51 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)

- [linear\\_probe\\_fn\\_imp.hpp](#)

#### 4.343 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

Inherits `type< Key, Mapped, _Alloc, list_update_tag, __gnu_cxx::typelist::create2< Eq_Fn, Update_Policy >::type >`.

##### Public Types

- typedef `list_update_tag` `container_category`
- typedef `Eq_Fn` `eq_fn`
- typedef `Update_Policy` `update_policy`

##### Public Member Functions

- template<typename `It` >  
`list_update` (`It first`, `It last`)
- `list_update` (`const list_update` &`other`)
- `list_update` & `operator=` (`const list_update` &`other`)
- void `swap` (`list_update` &`other`)

##### 4.343.1 Detailed Description

template<typename `Key`, typename `Mapped`, class `Eq_Fn` = typename `detail::default_eq_fn<Key>::type`, class `Update_Policy` = `detail::default_update_policy::type`, class `_Alloc` = `std::allocator<char>>`> class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`

A list-update based associative container.

##### Template Parameters

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                         |
| <i>Mapped</i>        | Map type.                                                                         |
| <i>Eq_Fn</i>         | Equal functor.                                                                    |
| <i>Update_Policy</i> | Update policy, determines when an element will be moved to the front of the list. |
| <i>_Alloc</i>        | Allocator type.                                                                   |

Base is `detail::lu_map`.

Definition at line 815 of file `assoc_container.hpp`.

##### 4.343.2 Constructor & Destructor Documentation

4.343.2.1 template<typename `Key` , typename `Mapped` , class `Eq_Fn` = typename `detail::default_eq_fn<Key>::type`, class `Update_Policy` = `detail::default_update_policy::type`, class `_Alloc` = `std::allocator<char>>`> template<typename `It` > `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update ( It first, It last )` [inline]

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

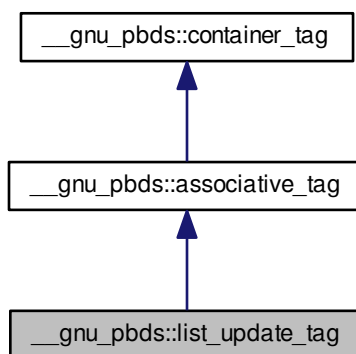
Definition at line 831 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

#### 4.344 \_\_gnu\_pbds::list\_update\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::list\_update\_tag:



##### 4.344.1 Detailed Description

List-update.

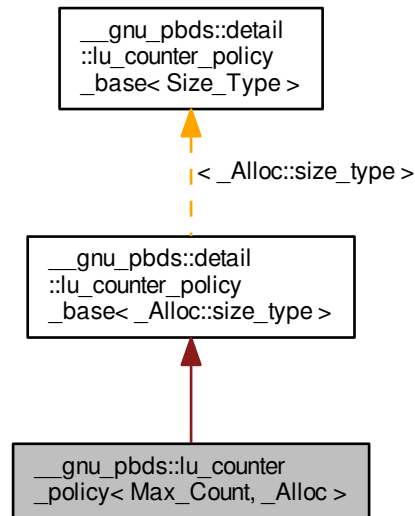
Definition at line 168 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.345 \_\_gnu\_pbds::lu\_counter\_policy&lt; Max\_Count, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >:



## Public Types

- enum { `max_count` }
- typedef `_Alloc allocator_type`
- typedef `__rebind_m::other::reference metadata_reference`
- typedef `detail::lu_counter_metadata< size_type > metadata_type`
- typedef `allocator_type::size_type size_type`

## Public Member Functions

- `metadata_type operator() () const`
- `bool operator() (metadata_reference r_data) const`

## Private Member Functions

- `lu_counter_metadata< size_type > operator() (size_type max_size) const`
- `bool operator() (Metadata_Reference r_data, size_type m_max_count) const`

## 4.345.1 Detailed Description

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

Definition at line 92 of file `list_update_policy.hpp`.

## 4.345.2 Member Typedef Documentation

```
4.345.2.1 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

Definition at line 115 of file `list_update_policy.hpp`.

```
4.345.2.2 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef
detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

Definition at line 107 of file `list_update_policy.hpp`.

## 4.345.3 Member Enumeration Documentation

```
4.345.3.1 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> anonymous enum
```

Enumerator:

***max\_count*** When some element is accessed this number of times, it will be moved to the front of the list.

Definition at line 99 of file `list_update_policy.hpp`.

## 4.345.4 Member Function Documentation

```
4.345.4.1 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> metadata_type
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() () const [inline]
```

Creates a metadata object.

Definition at line 119 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

```
4.345.4.2 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> bool __gnu_pbds-
::lu_counter_policy< Max_Count, _Alloc >::operator() (metadata_reference r.data) const
[inline]
```

Decides whether a metadata object should be moved to the front of the list.

Definition at line 125 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

## 4.346 `__gnu_pbds::lu_move_to_front_policy<_Alloc>` Class Template Reference

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__rebind_m::other::reference` [metadata\\_reference](#)
- typedef [null\\_type](#) **metadata\_type**

### Public Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) ([metadata\\_reference](#) r\_metadata) const

#### 4.346.1 Detailed Description

`template<typename _Alloc = std::allocator<char>> class __gnu_pbds::lu_move_to_front_policy<_Alloc>`

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

Definition at line 57 of file `list_update_policy.hpp`.

#### 4.346.2 Member Typedef Documentation

4.346.2.1 `template<typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference`  
`__gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_reference`

Reference to metadata on which this functor operates.

Definition at line 70 of file `list_update_policy.hpp`.

4.346.2.2 `template<typename _Alloc = std::allocator<char>> typedef null_type`  
`__gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_type`

Metadata on which this functor operates.

Definition at line 63 of file `list_update_policy.hpp`.

#### 4.346.3 Member Function Documentation

4.346.3.1 `template<typename _Alloc = std::allocator<char>> metadata_type`  
`__gnu_pbds::lu_move_to_front_policy<_Alloc>::operator() ( ) const` `[inline]`

Creates a metadata object.

Definition at line 74 of file `list_update_policy.hpp`.

4.346.3.2 `template<typename _Alloc = std::allocator<char>> bool __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator()( metadata_reference r_metadata ) const [inline]`

Decides whether a metadata object should be moved to the front of the list.

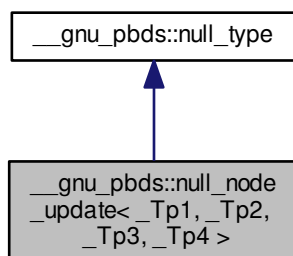
Definition at line 80 of file `list_update_policy.hpp`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

## 4.347 `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>`:



### 4.347.1 Detailed Description

`template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>struct __gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>`

A null node updatator, indicating that no node updates are required.

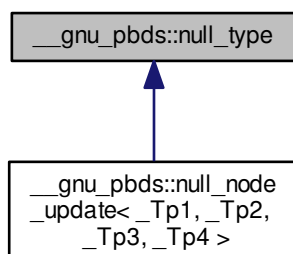
Definition at line 214 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.348 `__gnu_pbds::null_type` Struct Reference

Inheritance diagram for `__gnu_pbds::null_type`:



#### 4.348.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

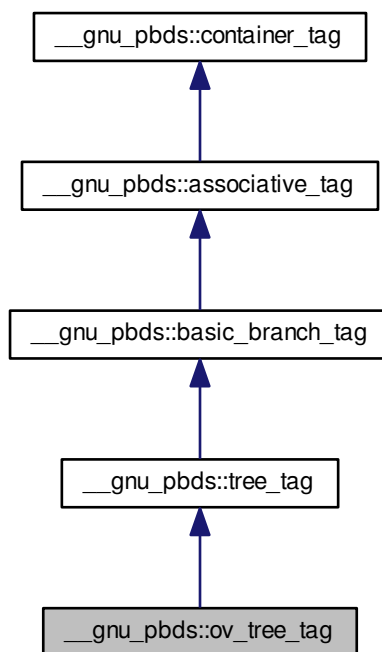
Definition at line 210 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.349 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::ov\_tree\_tag:



## 4.349.1 Detailed Description

Ordered-vector tree.

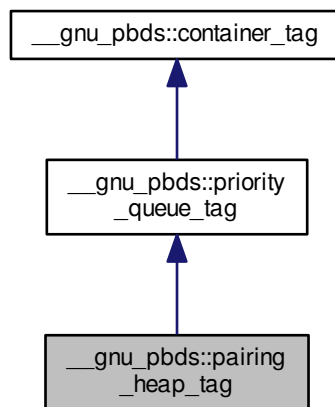
Definition at line 159 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.350 `__gnu_pbds::pairing_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::pairing_heap_tag`:



## 4.350.1 Detailed Description

Pairing-heap.

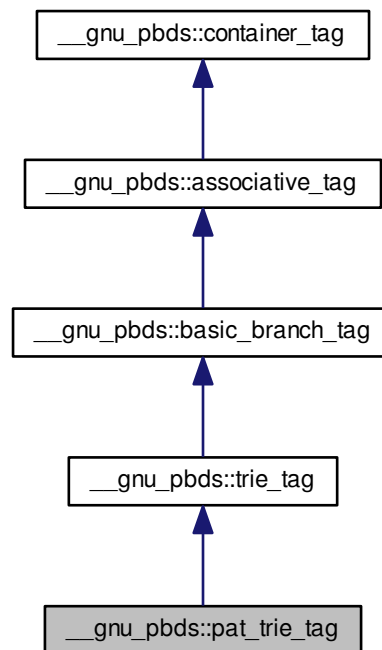
Definition at line 174 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.351 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



## 4.351.1 Detailed Description

PATRICIA trie.

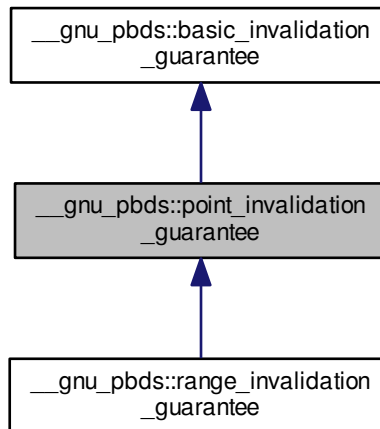
Definition at line 165 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.352 `__gnu_pbds::point_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::point_invalidation_guarantee`:



## 4.352.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.353 `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>` Class Template Reference

Inherits `type<_Tv, Cmp_Fn, _Alloc, Tag>`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `Tag` **container\_category**

- typedef `allocator_type::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `allocator_type::size_type` **size\_type**
- typedef `_Tv` **value\_type**

#### Public Member Functions

- [priority\\_queue](#) (const `cmp_fn` &`r_cmp_fn`)
- `template<typename It >`  
[priority\\_queue](#) (It `first_it`, It `last_it`)
- `template<typename It >`  
[priority\\_queue](#) (It `first_it`, It `last_it`, const `cmp_fn` &`r_cmp_fn`)
- **priority\_queue** (const [priority\\_queue](#) &`other`)
- [priority\\_queue](#) & **operator=** (const [priority\\_queue](#) &`other`)
- void **swap** ([priority\\_queue](#) &`other`)

#### 4.353.1 Detailed Description

`template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>` **class** `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>`

A priority queue composed of one specific heap policy.

#### Template Parameters

|                     |                                                                     |
|---------------------|---------------------------------------------------------------------|
| <code>_Tv</code>    | Value type.                                                         |
| <code>Cmp_Fn</code> | Comparison functor.                                                 |
| <code>Tag</code>    | Instantiating data structure type, see <code>container_tag</code> . |
| <code>_Alloc</code> | Allocator type.                                                     |

Base is dispatched at compile time via `Tag`, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.

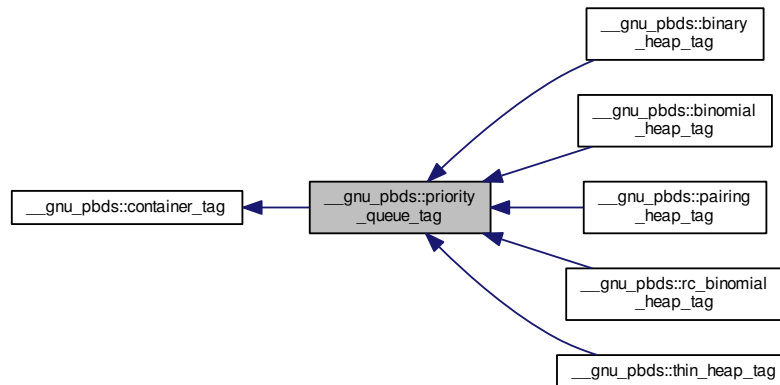
Definition at line 83 of file `priority_queue.hpp`.

The documentation for this class was generated from the following file:

- [priority\\_queue.hpp](#)

## 4.354 \_\_gnu\_pbds::priority\_queue\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::priority\_queue\_tag:



## 4.354.1 Detailed Description

Basic priority-queue.

Definition at line 171 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.355 \_\_gnu\_pbds::quadratic\_probe\_fn&lt; Size\_Type &gt; Class Template Reference

## Public Types

- typedef Size\_Type **size\_type**

## Public Member Functions

- void **swap** ([quadratic\\_probe\\_fn](#)< Size\_Type > &other)

## Protected Member Functions

- size\_type [operator\(\)](#) (size\_type i) const

## 4.355.1 Detailed Description

```
template<typename Size_Type = std::size_t>class __gnu_pbds::quadratic_probe_fn< Size_Type >
```

A probe sequence policy using square increments.

Definition at line 85 of file `hash_policy.hpp`.

#### 4.355.2 Member Function Documentation

4.355.2.1 `template<typename Size_Type > quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator() ( size_type i ) const` `[inline]`, `[protected]`

Returns the i-th offset from the hash value.

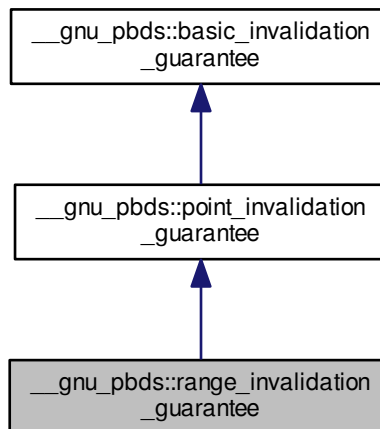
Definition at line 51 of file `hash_policy.hpp`.

The documentation for this class was generated from the following files:

- [hash\\_policy.hpp](#)
- [quadratic\\_probe\\_fn\\_imp.hpp](#)

#### 4.356 `__gnu_pbds::range_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::range_invalidation_guarantee`:



#### 4.356.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

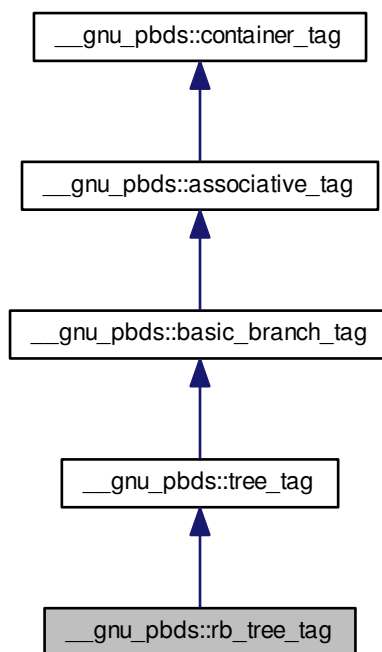
Definition at line 114 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.357 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::rb\_tree\_tag:



## 4.357.1 Detailed Description

Red-black tree.

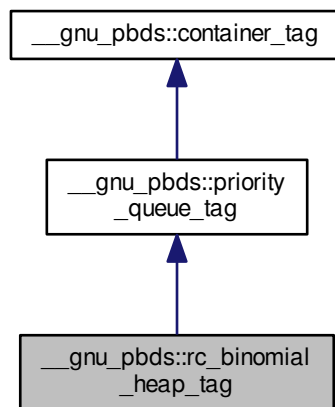
Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.358 `__gnu_pbds::rc_binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::rc_binomial_heap_tag`:



## 4.358.1 Detailed Description

Redundant-counter binomial-heap.

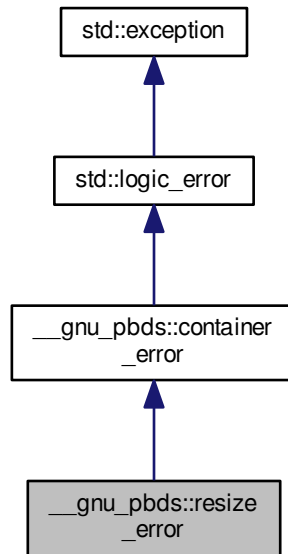
Definition at line 180 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.359 `__gnu_pbds::resize_error` Struct Reference

Inheritance diagram for `__gnu_pbds::resize_error`:



## Public Member Functions

- virtual const char \* [what](#) () const noexcept

## 4.359.1 Detailed Description

A container cannot be resized.

Definition at line 73 of file `exception.hpp`.

## 4.359.2 Member Function Documentation

4.359.2.1 `virtual const char* std::logic_error::what ( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.360 `__gnu_pbds::sample_probe_fn` Class Reference

## Public Types

- typedef `std::size_t` **size\_type**

## Public Member Functions

- [sample\\_probe\\_fn](#) ()
- [sample\\_probe\\_fn](#) (const [sample\\_probe\\_fn](#) &)
- void [swap](#) ([sample\\_probe\\_fn](#) &)

## Protected Member Functions

- `size_type` [operator\(\)](#) (key\_const\_reference *r\_key*, `size_type` *i*) const

## 4.360.1 Detailed Description

A sample probe policy.

Definition at line 47 of file `sample_probe_fn.hpp`.

## 4.360.2 Constructor &amp; Destructor Documentation

4.360.2.1 `__gnu_pbds::sample_probe_fn::sample_probe_fn ( )`

Default constructor.

4.360.2.2 `__gnu_pbds::sample_probe_fn::sample_probe_fn ( const sample\_probe\_fn & )`

Copy constructor.

## 4.360.3 Member Function Documentation

4.360.3.1 `size_type __gnu_pbds::sample_probe_fn::operator() ( key_const_reference r_key, size_type i ) const` `[inline]`, `[protected]`

Returns the *i*-th offset from the hash value of some key *r\_key*.

4.360.3.2 `void __gnu_pbds::sample_probe_fn::swap ( sample\_probe\_fn & )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_probe\\_fn.hpp](#)

4.361 `__gnu_pbds::sample_range_hashing` Class Reference

## Public Types

- typedef `std::size_t` [size\\_type](#)

## Public Member Functions

- [sample\\_range\\_hashing](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_range\\_hashing](#) &other)
- void [swap](#) ([sample\\_range\\_hashing](#) &other)

## Protected Member Functions

- void [notify\\_resized](#) ([size\\_type](#))
- [size\\_type](#) [operator\(\)](#) ([size\\_type](#)) const

### 4.361.1 Detailed Description

A sample range-hashing functor.

Definition at line 47 of file `sample_range_hashing.hpp`.

### 4.361.2 Member Typedef Documentation

#### 4.361.2.1 `typedef std::size_t __gnu_pbds::sample_range_hashing::size_type`

Size type.

Definition at line 51 of file `sample_range_hashing.hpp`.

### 4.361.3 Constructor & Destructor Documentation

#### 4.361.3.1 `__gnu_pbds::sample_range_hashing::sample_range_hashing ( )`

Default constructor.

#### 4.361.3.2 `__gnu_pbds::sample_range_hashing::sample_range_hashing ( const sample\_range\_hashing & other )`

Copy constructor.

### 4.361.4 Member Function Documentation

#### 4.361.4.1 `void __gnu_pbds::sample_range_hashing::notify_resized ( size\_type )` `[protected]`

Notifies the policy object that the container's size has changed to argument's size.

#### 4.361.4.2 `size\_type __gnu_pbds::sample_range_hashing::operator() ( size\_type ) const` `[inline]`, `[protected]`

Transforms the `__hash` value hash into a ranged-hash value.

#### 4.361.4.3 `void __gnu_pbds::sample_range_hashing::swap ( sample\_range\_hashing & other )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_range\\_hashing.hpp](#)

## 4.362 `__gnu_pbds::sample_ranged_hash_fn` Class Reference

### Public Types

- typedef `std::size_t` **size\_type**

### Public Member Functions

- [sample\\_ranged\\_hash\\_fn](#) ()
- [sample\\_ranged\\_hash\\_fn](#) (const [sample\\_ranged\\_hash\\_fn](#) &)
- void [swap](#) ([sample\\_ranged\\_hash\\_fn](#) &)

### Protected Member Functions

- void [notify\\_resized](#) (size\_type)
- size\_type [operator\(\)](#) (key\_const\_reference) const

#### 4.362.1 Detailed Description

A sample ranged-hash functor.

Definition at line 47 of file `sample_ranged_hash_fn.hpp`.

#### 4.362.2 Constructor & Destructor Documentation

##### 4.362.2.1 `__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ( )`

Default constructor.

##### 4.362.2.2 `__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ( const sample\_ranged\_hash\_fn & )`

Copy constructor.

#### 4.362.3 Member Function Documentation

##### 4.362.3.1 `void __gnu_pbds::sample_ranged_hash_fn::notify_resized ( size_type )` `[protected]`

Notifies the policy object that the container's `__size` has changed to `size`.

##### 4.362.3.2 `size_type __gnu_pbds::sample_ranged_hash_fn::operator() ( key_const_reference ) const` `[inline]`, `[protected]`

Transforms `key_const_reference` into a position within the table.

##### 4.362.3.3 `void __gnu_pbds::sample_ranged_hash_fn::swap ( sample\_ranged\_hash\_fn & )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_hash\\_fn.hpp](#)

## 4.363 `__gnu_pbds::sample_ranged_probe_fn` Class Reference

### Public Types

- typedef std::size\_t **size\_type**

### Public Member Functions

- **sample\_ranged\_probe\_fn** (const [sample\\_ranged\\_probe\\_fn](#) &)
- void **swap** ([sample\\_ranged\\_probe\\_fn](#) &)

### Protected Member Functions

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference, std::size\_t, size\_type) const

#### 4.363.1 Detailed Description

A sample ranged-probe functor.

Definition at line 47 of file [sample\\_ranged\\_probe\\_fn.hpp](#).

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_probe\\_fn.hpp](#)

## 4.364 `__gnu_pbds::sample_resize_policy` Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_policy](#) &other)
- void **swap** ([sample\\_resize\\_policy](#) &other)

### Protected Member Functions

- [size\\_type](#) **get\_new\_size** ([size\\_type](#) size, [size\\_type](#) num\_used\_e) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** ([size\\_type](#) num\_e)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()

- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (`size_type` num\_e)
- void `notify_resized` (`size_type` new\_size)

#### 4.364.1 Detailed Description

A sample resize policy.

Definition at line 47 of file `sample_resize_policy.hpp`.

#### 4.364.2 Member Typedef Documentation

##### 4.364.2.1 `typedef std::size_t __gnu_pbds::sample_resize_policy::size_type`

Size type.

Definition at line 51 of file `sample_resize_policy.hpp`.

#### 4.364.3 Constructor & Destructor Documentation

##### 4.364.3.1 `__gnu_pbds::sample_resize_policy::sample_resize_policy` ( )

Default constructor.

#### 4.364.4 Member Function Documentation

##### 4.364.4.1 `size_type __gnu_pbds::sample_resize_policy::get_new_size` ( `size_type` size, `size_type` num\_used\_e ) const [protected]

Queries what the new size should be.

##### 4.364.4.2 `bool __gnu_pbds::sample_resize_policy::is_resize_needed` ( ) const [inline], [protected]

Queries whether a resize is needed.

##### 4.364.4.3 `void __gnu_pbds::sample_resize_policy::notify_cleared` ( ) [protected]

Notifies the table was cleared.

##### 4.364.4.4 `void __gnu_pbds::sample_resize_policy::notify_erase_search_collision` ( ) [inline], [protected]

Notifies a search encountered a collision.

##### 4.364.4.5 `void __gnu_pbds::sample_resize_policy::notify_erase_search_end` ( ) [inline], [protected]

Notifies a search ended.

##### 4.364.4.6 `void __gnu_pbds::sample_resize_policy::notify_erase_search_start` ( ) [inline], [protected]

Notifies a search started.

4.364.4.7 `void __gnu_pbds::sample_resize_policy::notify_erased ( size_type num_e )` `[inline]`, `[protected]`

Notifies an element was erased.

4.364.4.8 `void __gnu_pbds::sample_resize_policy::notify_find_search_collision ( )` `[inline]`, `[protected]`

Notifies a search encountered a collision.

4.364.4.9 `void __gnu_pbds::sample_resize_policy::notify_find_search_end ( )` `[inline]`, `[protected]`

Notifies a search ended.

4.364.4.10 `void __gnu_pbds::sample_resize_policy::notify_find_search_start ( )` `[inline]`, `[protected]`

Notifies a search started.

4.364.4.11 `void __gnu_pbds::sample_resize_policy::notify_insert_search_collision ( )` `[inline]`, `[protected]`

Notifies a search encountered a collision.

4.364.4.12 `void __gnu_pbds::sample_resize_policy::notify_insert_search_end ( )` `[inline]`, `[protected]`

Notifies a search ended.

4.364.4.13 `void __gnu_pbds::sample_resize_policy::notify_insert_search_start ( )` `[inline]`, `[protected]`

Notifies a search started.

4.364.4.14 `void __gnu_pbds::sample_resize_policy::notify_inserted ( size_type num_e )` `[inline]`, `[protected]`

Notifies an element was inserted.

4.364.4.15 `void __gnu_pbds::sample_resize_policy::notify_resized ( size_type new_size )` `[protected]`

Notifies the table was resized to `new_size`.

4.364.4.16 `__gnu_pbds::sample_resize_policy::sample_range_hashing ( const sample_resize_policy & other )`

Copy constructor.

4.364.4.17 `void __gnu_pbds::sample_resize_policy::swap ( sample_resize_policy & other )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_policy.hpp](#)

## 4.365 `__gnu_pbds::sample_resize_trigger` Class Reference

### Public Types

- `typedef std::size_t` [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_trigger](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_trigger](#) &)
- void [swap](#) ([sample\\_resize\\_trigger](#) &)

### Protected Member Functions

- bool [is\\_grow\\_needed](#) ([size\\_type](#) size, [size\\_type](#) num\_entries) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_entries)
- void [notify\\_externally\\_resized](#) ([size\\_type](#) new\_size)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_entries)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

#### 4.365.1 Detailed Description

A sample resize trigger policy.

Definition at line 47 of file `sample_resize_trigger.hpp`.

#### 4.365.2 Member Typedef Documentation

##### 4.365.2.1 `typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type`

Size type.

Definition at line 51 of file `sample_resize_trigger.hpp`.

#### 4.365.3 Constructor & Destructor Documentation

##### 4.365.3.1 `__gnu_pbds::sample_resize_trigger::sample_resize_trigger ( )`

Default constructor.

#### 4.365.4 Member Function Documentation

##### 4.365.4.1 `bool __gnu_pbds::sample_resize_trigger::is_grow_needed ( size_type size, size_type num_entries ) const` [inline], [protected]

Queries whether a grow is needed.

4.365.4.2 `bool __gnu_pbds::sample_resize_trigger::is_resize_needed ( ) const` [inline],[protected]

Queries whether a resize is needed.

4.365.4.3 `void __gnu_pbds::sample_resize_trigger::notify_cleared ( )` [protected]

Notifies the table was cleared.

4.365.4.4 `void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision ( )` [inline],[protected]

Notifies a search encountered a collision.

4.365.4.5 `void __gnu_pbds::sample_resize_trigger::notify_erase_search_end ( )` [inline],[protected]

Notifies a search ended.

4.365.4.6 `void __gnu_pbds::sample_resize_trigger::notify_erase_search_start ( )` [inline],[protected]

Notifies a search started.

4.365.4.7 `void __gnu_pbds::sample_resize_trigger::notify_erased ( size_type num_entries )` [inline],[protected]

Notifies an element was erased.

4.365.4.8 `void __gnu_pbds::sample_resize_trigger::notify_externally_resized ( size_type new_size )` [protected]

Notifies the table was resized externally.

4.365.4.9 `void __gnu_pbds::sample_resize_trigger::notify_find_search_collision ( )` [inline],[protected]

Notifies a search encountered a collision.

4.365.4.10 `void __gnu_pbds::sample_resize_trigger::notify_find_search_end ( )` [inline],[protected]

Notifies a search ended.

4.365.4.11 `void __gnu_pbds::sample_resize_trigger::notify_find_search_start ( )` [inline],[protected]

Notifies a search started.

4.365.4.12 `void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision ( )` [inline],[protected]

Notifies a search encountered a collision.

4.365.4.13 `void __gnu_pbds::sample_resize_trigger::notify_insert_search_end ( )` [inline],[protected]

Notifies a search ended.

4.365.4.14 `void __gnu_pbds::sample_resize_trigger::notify_insert_search_start ( )` [inline],[protected]

Notifies a search started.

4.365.4.15 `void __gnu_pbds::sample_resize_trigger::notify_inserted ( size_type num_entries )` [inline],[protected]

Notifies an element was inserted. the total number of entries in the table is num\_entries.

4.365.4.16 `void __gnu_pbds::sample_resize_trigger::notify_resized ( size_type new_size )` `[protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

4.365.4.17 `__gnu_pbds::sample_resize_trigger::sample_range_hashing ( const sample_resize_trigger & )`

Copy constructor.

4.365.4.18 `void __gnu_pbds::sample_resize_trigger::swap ( sample_resize_trigger & )` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_trigger.hpp](#)

## 4.366 `__gnu_pbds::sample_size_policy` Class Reference

### Public Types

- `typedef std::size_t size_type`

### Public Member Functions

- `sample_size_policy ()`
- `sample_range_hashing (const sample_size_policy &)`
- `void swap (sample_size_policy &other)`

### Protected Member Functions

- `size_type get_nearest_larger_size (size_type size) const`
- `size_type get_nearest_smaller_size (size_type size) const`

#### 4.366.1 Detailed Description

A sample size policy.

Definition at line 47 of file `sample_size_policy.hpp`.

#### 4.366.2 Member Typedef Documentation

4.366.2.1 `typedef std::size_t __gnu_pbds::sample_size_policy::size_type`

Size type.

Definition at line 51 of file `sample_size_policy.hpp`.

#### 4.366.3 Constructor & Destructor Documentation

4.366.3.1 `__gnu_pbds::sample_size_policy::sample_size_policy ( )`

Default constructor.

#### 4.366.4 Member Function Documentation

4.366.4.1 `size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size ( size_type size ) const` [inline], [protected]

Given a `__size` size, returns a `__size` that is larger.

4.366.4.2 `size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size ( size_type size ) const` [inline], [protected]

Given a `__size` size, returns a `__size` that is smaller.

4.366.4.3 `__gnu_pbds::sample_size_policy::sample_range_hashing ( const sample_size_policy & )`

Copy constructor.

4.366.4.4 `void __gnu_pbds::sample_size_policy::swap ( sample_size_policy & other )` [inline]

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_size\\_policy.hpp](#)

### 4.367 `__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >` Class Template Reference

#### 4.367.1 Detailed Description

`template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc> class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >`

A sample node updatator.

Definition at line 49 of file `sample_tree_node_update.hpp`.

The documentation for this class was generated from the following file:

- [sample\\_tree\\_node\\_update.hpp](#)

### 4.368 `__gnu_pbds::sample_trie_access_traits` Struct Reference

#### Public Types

- enum { **max\_size** }
- typedef `_Alloc::template rebind< key_type > __rebind_k`
- typedef `std::string::const_iterator` **const\_iterator**
- typedef char **e\_type**
- typedef `__rebind_k::other::const_reference` **key\_const\_reference**
- typedef `std::string` **key\_type**
- typedef `std::size_t` **size\_type**

#### Static Public Member Functions

- static const\_iterator [begin](#) (key\_const\_reference)
- static size\_type [e\\_pos](#) (e\_type)
- static const\_iterator [end](#) (key\_const\_reference)

#### 4.368.1 Detailed Description

A sample trie element access traits.

Definition at line 47 of file `sample_trie_access_traits.hpp`.

#### 4.368.2 Member Typedef Documentation

##### 4.368.2.1 typedef char `__gnu_pbds::sample_trie_access_traits::e_type`

Element type.

Definition at line 57 of file `sample_trie_access_traits.hpp`.

#### 4.368.3 Member Function Documentation

##### 4.368.3.1 static const\_iterator `__gnu_pbds::sample_trie_access_traits::begin` ( key\_const\_reference ) `[inline]`, `[static]`

Returns a const\_iterator to the first element of r\_key.

##### 4.368.3.2 static size\_type `__gnu_pbds::sample_trie_access_traits::e_pos` ( e\_type ) `[inline]`, `[static]`

Maps an element to a position.

##### 4.368.3.3 static const\_iterator `__gnu_pbds::sample_trie_access_traits::end` ( key\_const\_reference ) `[inline]`, `[static]`

Returns a const\_iterator to the after-last element of r\_key.

The documentation for this struct was generated from the following file:

- [sample\\_trie\\_access\\_traits.hpp](#)

#### 4.369 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

#### Public Types

- typedef std::size\_t **metadata\_type**

#### Protected Member Functions

- [sample\\_trie\\_node\\_update](#) ()
- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

## 4.369.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::sample_trie_node_
update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

Definition at line 49 of file `sample_trie_node_update.hpp`.

## 4.369.2 Constructor &amp; Destructor Documentation

```
4.369.2.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_update
() [protected]
```

Default constructor.

## 4.369.3 Member Function Documentation

```
4.369.3.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (node_iterator ,
node_const_iterator) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

The documentation for this class was generated from the following file:

- [sample\\_trie\\_node\\_update.hpp](#)

4.370 `__gnu_pbds::sample_update_policy` Struct Reference

## Public Member Functions

- [sample\\_update\\_policy](#) ()
- [sample\\_update\\_policy](#) (const [sample\\_update\\_policy](#) &)
- void [swap](#) ([sample\\_update\\_policy](#) &other)

## Protected Types

- typedef some\_metadata\_type [metadata\\_type](#)

## Protected Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) (metadata\_reference) const

## 4.370.1 Detailed Description

A sample list-update policy.

Definition at line 47 of file `sample_update_policy.hpp`.

#### 4.370.2 Member Typedef Documentation

##### 4.370.2.1 `typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type` `[protected]`

Metadata on which this functor operates.

Definition at line 61 of file `sample_update_policy.hpp`.

#### 4.370.3 Constructor & Destructor Documentation

##### 4.370.3.1 `__gnu_pbds::sample_update_policy::sample_update_policy ( )`

Default constructor.

##### 4.370.3.2 `__gnu_pbds::sample_update_policy::sample_update_policy ( const sample_update_policy & )`

Copy constructor.

#### 4.370.4 Member Function Documentation

##### 4.370.4.1 `metadata_type __gnu_pbds::sample_update_policy::operator()( ) const` `[protected]`

Creates a metadata object.

##### 4.370.4.2 `bool __gnu_pbds::sample_update_policy::operator()( metadata_reference ) const` `[protected]`

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

##### 4.370.4.3 `void __gnu_pbds::sample_update_policy::swap ( sample_update_policy & other )` `[inline]`

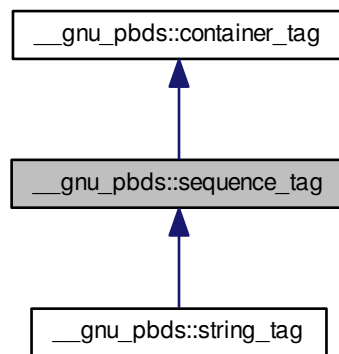
Swaps content.

The documentation for this struct was generated from the following file:

- [sample\\_update\\_policy.hpp](#)

## 4.371 \_\_gnu\_pbds::sequence\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



## 4.371.1 Detailed Description

Basic sequence.

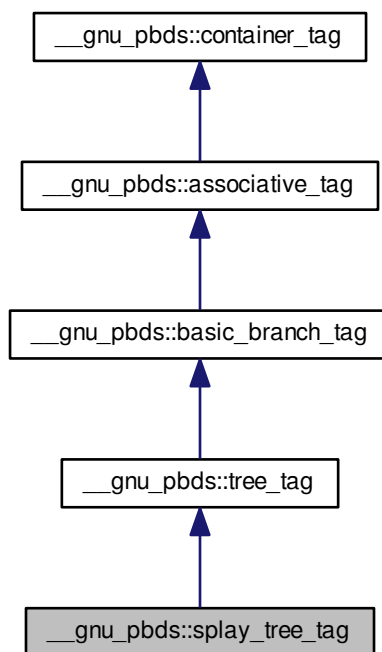
Definition at line 129 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.372 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::splay\_tree\_tag:



## 4.372.1 Detailed Description

Splay tree.

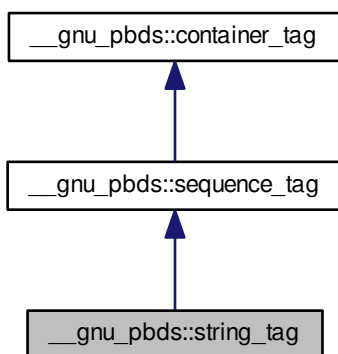
Definition at line 156 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.373 `__gnu_pbds::string_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::string_tag`:



## 4.373.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

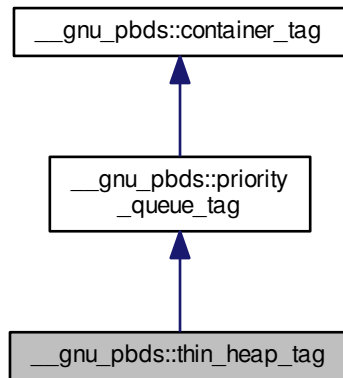
Definition at line 132 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.374 \_\_gnu\_pbds::thin\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::thin\_heap\_tag:



## 4.374.1 Detailed Description

Thin heap.

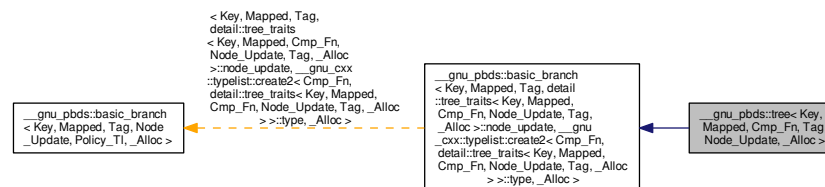
Definition at line 186 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.375 \_\_gnu\_pbds::tree&lt; Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >:



## Public Types

- typedef Cmp\_Fn [cmp\\_fn](#)

- typedef detail::tree\_traits  
`< Key, Mapped, Cmp_Fn, Node_Update, Tag, _Alloc >`  
`::node_update` **node\_update**

#### Public Member Functions

- `tree` (const `cmp_fn` &c)
- template<typename It >  
`tree` (It first, It last)
- template<typename It >  
`tree` (It first, It last, const `cmp_fn` &c)
- **tree** (const `tree` &other)
- `tree` & **operator=** (const `tree` &other)
- void **swap** (`tree` &other)

#### 4.375.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp\_Fn = std::less<Key>, typename Tag = rb\_tree\_tag, template< typename Node\_Cltr, typename Node\_Itr, typename Cmp\_Fn\_, typename \_Alloc\_ > class Node\_Update = null\_node\_update, typename \_Alloc = std::allocator<char>> class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`

A tree-based container.

#### Template Parameters

|                    |                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                |
| <i>Mapped</i>      | Map type.                                                                                                                |
| <i>Cmp_Fn</i>      | Comparison functor.                                                                                                      |
| <i>Tag</i>         | Instantiating data structure type, see container_tag.                                                                    |
| <i>Node_Update</i> | Updates tree internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                          |

Base tag choices are: `ov_tree_tag`, `rb_tree_tag`, `splay_tree_tag`.

Base is `basic_branch`.

Definition at line 635 of file `assoc_container.hpp`.

#### 4.375.2 Member Typedef Documentation

4.375.2.1 template<typename Key , typename Mapped , typename Cmp\_Fn = std::less<Key>, typename Tag = rb\_tree\_tag, template< typename Node\_Cltr, typename Node\_Itr, typename Cmp\_Fn\_, typename \_Alloc\_ > class Node\_Update = null\_node\_update, typename \_Alloc = std::allocator<char>> typedef Cmp\_Fn `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn`

Comparison functor type.

Definition at line 642 of file `assoc_container.hpp`.

#### 4.375.3 Constructor & Destructor Documentation

4.375.3.1 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree ( const cmp_fn & c ) [inline]`

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

Definition at line 648 of file `assoc_container.hpp`.

4.375.3.2 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree ( It first, It last ) [inline]`

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 655 of file `assoc_container.hpp`.

4.375.3.3 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree ( It first, It last, const cmp_fn & c ) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object.

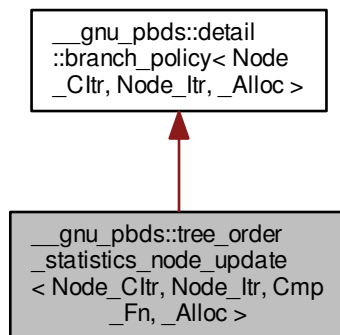
Definition at line 663 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 4.376 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc > Class` Template Reference

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef  
    `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef  
    `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

### Public Member Functions

- `const_iterator` [find\\_by\\_order](#) (`size_type`) const
- `iterator` [find\\_by\\_order](#) (`size_type`)
- `size_type` [order\\_of\\_key](#) (`key_const_reference`) const

### Protected Member Functions

- void [operator\(\)](#) (`node_iterator`, `node_const_iterator`) const

### Private Types

- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

### Private Member Functions

- virtual `it_type` **end** ()=0
- `it_type` **end\_iterator** () const

### Static Private Member Functions

- static `key_const_reference` **extract\_key** (`const_reference` r\_val)

#### 4.376.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`

Functor updating ranks of entrees.

Definition at line 64 of file `tree_policy.hpp`.

#### 4.376.2 Member Function Documentation

4.376.2.1 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >`  
`tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator`  
`__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (`  
`size_type order ) const [inline]`

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 72 of file `tree_policy.hpp`.

4.376.2.2 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >`  
`tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::iterator`  
`__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (`  
`size_type order ) [inline]`

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file `tree_policy.hpp`.

4.376.2.3 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > void`  
`__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::operator() (`  
`node_iterator node_it, node_const_iterator end_nd_it ) const [inline], [protected]`

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 108 of file `tree_policy.hpp`.

4.376.2.4 `template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >`  
`tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::size_type`  
`__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (`  
`key_const_reference r_key ) const [inline]`

Returns the order of a key within a sequence. For exapmle, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

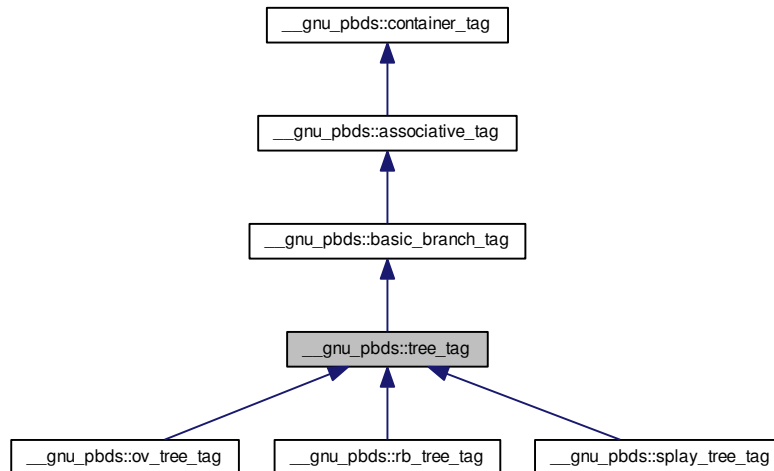
Definition at line 78 of file `tree_policy.hpp`.

The documentation for this class was generated from the following files:

- [tree\\_policy.hpp](#)
- [tree\\_policy/order\\_statistics\\_imp.hpp](#)

## 4.377 \_\_gnu\_pbds::tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::tree\_tag:



## 4.377.1 Detailed Description

Basic tree structure.

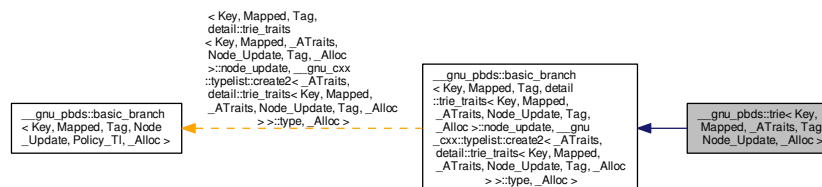
Definition at line 150 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.378 \_\_gnu\_pbds::trie&lt; Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >:



## Public Types

- typedef \_ATraits [access\\_traits](#)

- `typedef detail::trie_traits< Key, Mapped, _ATraits, Node_Update, Tag, _Alloc >::node_update node_update`

#### Public Member Functions

- `trie` (const `access_traits` &t)
- `template<typename It > trie` (It first, It last)
- `template<typename It > trie` (It first, It last, const `access_traits` &t)
- `trie` (const `trie` &other)
- `trie & operator=` (const `trie` &other)
- `void swap` (`trie` &other)

#### 4.378.1 Detailed Description

`template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`

A trie-based container.

#### Template Parameters

|                    |                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                              |
| <i>Mapped</i>      | Map type.                                                                                                                              |
| <i>_ATraits</i>    | Element access traits.                                                                                                                 |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> .                                                                    |
| <i>Node_Update</i> | Updates trie internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containers</code> node invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                                        |

Base tag choice is `pat_trie_tag`.

Base is `basic_branch`.

Definition at line 731 of file `assoc_container.hpp`.

#### 4.378.2 Member Typedef Documentation

4.378.2.1 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits`

Element access traits type.

Definition at line 738 of file `assoc_container.hpp`.

## 4.378.3 Constructor &amp; Destructor Documentation

4.378.3.1 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>-  
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename  
_Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> __gnu_pbds::trie< Key,  
Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie ( const access_traits & t ) [inline]`

Constructor taking some policy objects. `r_access_traits` will be copied by the `_ATraits` object of the container object.

Definition at line 744 of file `assoc_container.hpp`.

4.378.3.2 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>-  
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename  
_Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It >  
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie ( It first, It last ) [inline]`

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 751 of file `assoc_container.hpp`.

4.378.3.3 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>-  
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename  
_Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It >  
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie ( It first, It last, const access_traits & t  
) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

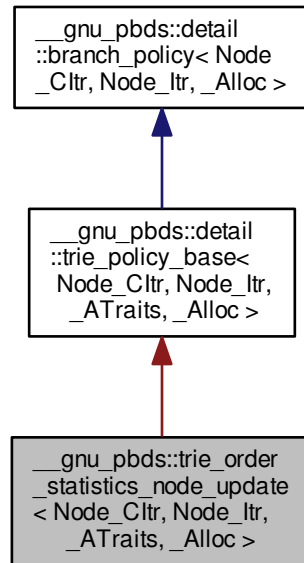
Definition at line 758 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.379 `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template  
Reference

Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



#### Public Types

- typedef `access_traits::const_iterator` **a\_const\_iterator**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

#### Public Member Functions

- `const_iterator` [find\\_by\\_order](#) (`size_type`) const
- `iterator` [find\\_by\\_order](#) (`size_type`)

- size\_type `order_of_key` (key\_const\_reference) const
- size\_type `order_of_prefix` (a\_const\_iterator, a\_const\_iterator) const

#### Protected Member Functions

- void `operator()` (node\_iterator, node\_const\_iterator) const

#### Private Types

- typedef Node\_Itr::value\_type **it\_type**
- typedef remove\_const< key\_type >::type **rckey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef \_Alloc::template rebind< rckey\_type >::other **rebind\_k**
- typedef \_Alloc::template rebind< rcvalue\_type >::other **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef std::iterator\_traits< it\_type >::value\_type **value\_type**

#### Private Member Functions

- virtual const\_iterator **end** () const =0
- it\_type **end\_iterator** () const
- virtual const access\_traits & **get\_access\_traits** () const =0

#### Static Private Member Functions

- static size\_type **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static iterator **rightmost\_it** (node\_iterator)

#### 4.379.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`

Functor updating ranks of entrees.

Definition at line 253 of file `trie_policy.hpp`.

#### 4.379.2 Member Function Documentation

4.379.2.1 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >  
 trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator  
 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::find_by_order (`  
`size_type order ) const [inline]`

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 79 of file `trie_policy.hpp`.

4.379.2.2 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >  
 trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator  
 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::find_by_order (`  
`size_type order ) [inline]`

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file `trie_policy.hpp`.

4.379.2.3 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void  
 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (`  
`node_iterator nd_it, node_const_iterator ) const [inline], [protected]`

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 152 of file `trie_policy.hpp`.

4.379.2.4 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >  
 trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type  
 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_key (`  
`key_const_reference r_key ) const [inline]`

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

Definition at line 85 of file `trie_policy.hpp`.

4.379.2.5 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >  
 trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type  
 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (`  
`a_const_iterator b, a_const_iterator e ) const [inline]`

Returns the order of a prefix within a sequence. For example, if `[b, e]` is the smallest prefix, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

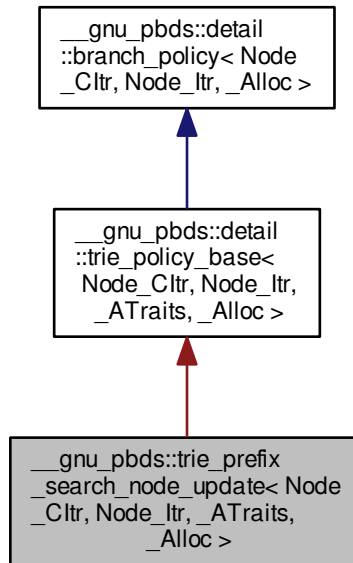
Definition at line 96 of file `trie_policy.hpp`.

The documentation for this class was generated from the following files:

- [trie\\_policy.hpp](#)
- [trie\\_policy/order\\_statistics\\_imp.hpp](#)

## 4.380 `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class` Template Reference

Inheritance diagram for `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



### Public Types

- typedef `access_traits::const_iterator` [a\\_const\\_iterator](#)
- typedef `_ATraits` [access\\_traits](#)
- typedef `_Alloc` [allocator\\_type](#)
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef [null\\_type](#) **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` [size\\_type](#)

### Public Member Functions

- [std::pair](#)< `const_iterator`, `const_iterator` > [prefix\\_range](#) (`key_const_reference`) const

- `std::pair< iterator, iterator > prefix_range` (key\_const\_reference)
- `std::pair< const_iterator, const_iterator > prefix_range` (a\_const\_iterator, a\_const\_iterator) const
- `std::pair< iterator, iterator > prefix_range` (a\_const\_iterator, a\_const\_iterator)

#### Protected Member Functions

- void `operator()` (node\_iterator node\_it, node\_const\_iterator end\_nd\_it) const

#### Private Types

- typedef rebind\_v::const\_pointer **const\_pointer**
- typedef rebind\_v::const\_reference **const\_reference**
- typedef Node\_Itr::value\_type **it\_type**
- typedef remove\_const< key\_type >::type **rckey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef \_Alloc::template rebind< rckey\_type >::other **rebind\_k**
- typedef \_Alloc::template rebind< rcvalue\_type >::other **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef std::iterator\_traits< it\_type >::value\_type **value\_type**

#### Private Member Functions

- it\_type **end\_iterator** () const

#### Static Private Member Functions

- static `size_type common_prefix_len` (node\_iterator, e\_const\_iterator, e\_const\_iterator, const `access_traits` &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const `access_traits` &)
- static iterator **rightmost\_it** (node\_iterator)

#### 4.380.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`

A node updatator that allows tries to be searched for the range of values that match a certain prefix.

Definition at line 155 of file `trie_policy.hpp`.

#### 4.380.2 Member Typedef Documentation

4.380.2.1 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef  
access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc  
>::a_const_iterator`

Const element iterator.

Definition at line 168 of file `trie_policy.hpp`.

4.380.2.2 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _ATraits  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::access_traits`

Element access traits.

Definition at line 165 of file `trie_policy.hpp`.

4.380.2.3 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _Alloc  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::allocator_type`

`_Alloc` type.

Definition at line 171 of file `trie_policy.hpp`.

4.380.2.4 `template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef allocator_type::size_type  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type`

Size type.

Definition at line 174 of file `trie_policy.hpp`.

#### 4.380.3 Member Function Documentation

4.380.3.1 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (  
node_iterator node_it, node_const_iterator end_nd_it ) const [inline], [protected]`

Called to update a node's metadata.

Definition at line 139 of file `trie_policy.hpp`.

4.380.3.2 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator,  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator >  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (  
key_const_reference r_key ) const`

Finds the const iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 47 of file `trie_policy.hpp`.

4.380.3.3 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator,  
typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator >  
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (  
key_const_reference r_key )`

Finds the iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 58 of file `trie_policy.hpp`.

```
4.380.3.4 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<
 typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator,
 typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator >
 __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (
 a_const_iterator b, a_const_iterator e) const
```

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 69 of file `trie_policy.hpp`.

```
4.380.3.5 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<
 typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator,
 typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator >
 __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (
 a_const_iterator b, a_const_iterator e)
```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 84 of file `trie_policy.hpp`.

The documentation for this class was generated from the following files:

- [trie\\_policy.hpp](#)
- [prefix\\_search\\_node\\_update\\_imp.hpp](#)

## 4.381 `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc > Struct` Template Reference

### Public Types

- enum { **reverse** }
- enum { **min\_e\_val**, **max\_e\_val**, **max\_size** }
- typedef `_Alloc::template rebind< key_type > __rebind_k`
- typedef `detail::__conditional_type< Reverse, typename String::const_reverse_iterator, typename String::const_iterator >::__type` [const\\_iterator](#)
- typedef `std::iterator_traits< const\_iterator >::value_type` [e\\_type](#)
- typedef `__rebind_k::other::const_reference` **key\_const\_reference**
- typedef `String` **key\_type**
- typedef `_Alloc::size_type` **size\_type**

### Static Public Member Functions

- static [const\\_iterator](#) [begin](#) (key\_const\_reference)
- static `size_type` [e\\_pos](#) ([e\\_type](#) e)
- static [const\\_iterator](#) [end](#) (key\_const\_reference)

## 4.381.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

## Template Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>String</i>    | String type.                                      |
| <i>Min_E_Val</i> | Minimal element value.                            |
| <i>Max_E_Val</i> | Maximum element value.                            |
| <i>Reverse</i>   | Reverse iteration should be used. Default: false. |
| <i>_Alloc</i>    | Allocator type.                                   |

Definition at line 74 of file `trie_policy.hpp`.

## 4.381.2 Member Typedef Documentation

4.381.2.1 `template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef detail::__conditional_type<Reverse, typename String::const_reverse_iterator, typename String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator`

Element const iterator type.

Definition at line 90 of file `trie_policy.hpp`.

4.381.2.2 `template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type`

Element type.

Definition at line 93 of file `trie_policy.hpp`.

## 4.381.3 Member Function Documentation

4.381.3.1 `template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin ( key_const_reference r.key ) [inline], [static]`

Returns a `const_iterator` to the first element of `key_const_reference` agumnet.

Definition at line 57 of file `trie_policy.hpp`.

```
4.381.3.2 template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool
Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (e_type e)
[inline], [static]
```

Maps an element to a position.

Definition at line 49 of file trie\_policy.hpp.

```
4.381.3.3 template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val,
bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc
>::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc
>::end (key_const_reference r.key) [inline], [static]
```

Returns a const\_iterator to the after-last element of key\_const\_reference argument.

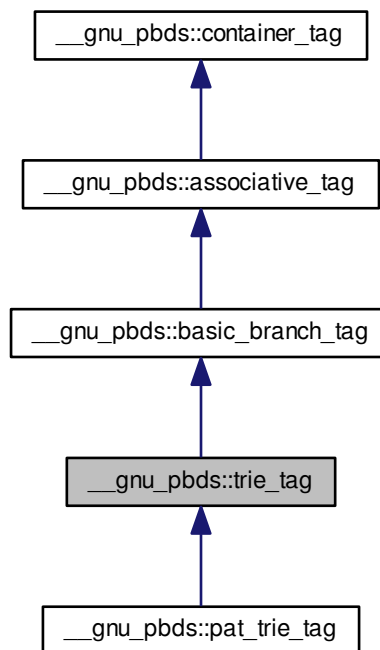
Definition at line 65 of file trie\_policy.hpp.

The documentation for this struct was generated from the following files:

- [trie\\_policy.hpp](#)
- [trie\\_string\\_access\\_traits\\_imp.hpp](#)

## 4.382 \_\_gnu\_pbds::trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::trie\_tag:



## 4.382.1 Detailed Description

Basic trie structure.

Definition at line 162 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.383 `__gnu_pbds::trivial_iterator_tag` Struct Reference

## 4.383.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of `std::iterators`'s movement abilities.

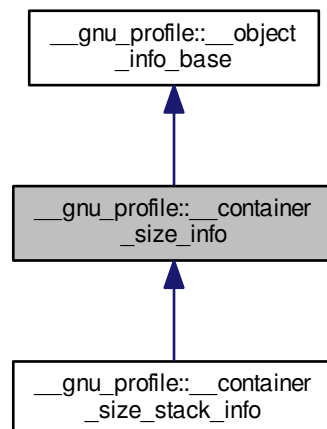
Definition at line 75 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.384 `__gnu_profile::__container_size_info` Class Reference

Inheritance diagram for `__gnu_profile::__container_size_info`:



## Public Member Functions

- `__container_size_info` (const [\\_\\_container\\_size\\_info](#) &`_o`)
- `__container_size_info` (`__stack_t` `_stack`, `std::size_t` `_num`)
- `std::string` `__advice` () const

- void **\_\_destruct** (std::size\_t \_\_num, std::size\_t \_\_inum)
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- void **\_\_resize** (std::size\_t \_\_from, std::size\_t \_\_to)
- float **\_\_resize\_cost** (std::size\_t \_\_from, std::size\_t)
- `__stack_t` **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- `__stack_t` **M\_stack**
- bool **M\_valid**

#### 4.384.1 Detailed Description

A container size instrumentation line in the object table.

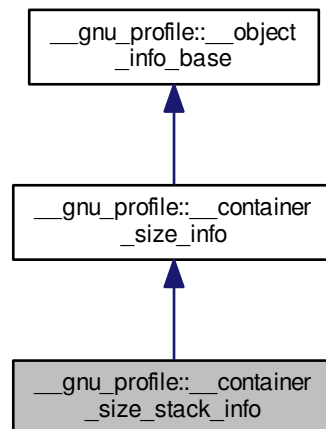
Definition at line 42 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

#### 4.385 `__gnu_profile::__container_size_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__container_size_stack_info`:



## Public Member Functions

- `__container_size_stack_info` (const `__container_size_info` &\_\_o)
- `std::string __advice` () const
- void `__destruct` (std::size\_t \_\_num, std::size\_t \_\_inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__container_size_info` &\_\_o)
- void `__resize` (std::size\_t \_\_from, std::size\_t \_\_to)
- float `__resize_cost` (std::size\_t \_\_from, std::size\_t)
- `__stack_t __stack` () const
- void `__write` (FILE \*\_\_f) const

## Protected Attributes

- `__stack_t _M_stack`
- bool `_M_valid`

## 4.385.1 Detailed Description

A container size instrumentation line in the stack table.

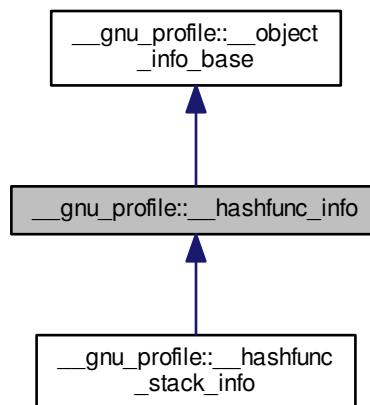
Definition at line 154 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

4.386 `__gnu_profile::__hashfunc_info` Class Reference

Inheritance diagram for `__gnu_profile::__hashfunc_info`:



## Public Member Functions

- `__hashfunc_info` (const `__hashfunc_info` &\_\_o)
- `__hashfunc_info` (`__stack_t` \_\_stack)
- `std::string` `__advice` () const
- void `__destruct` (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__hashfunc_info` &\_\_o)
- `__stack_t` `__stack` () const
- void `__write` (FILE \*\_\_f) const

## Protected Attributes

- `__stack_t` `_M_stack`
- bool `_M_valid`

## 4.386.1 Detailed Description

A hash performance instrumentation line in the object table.

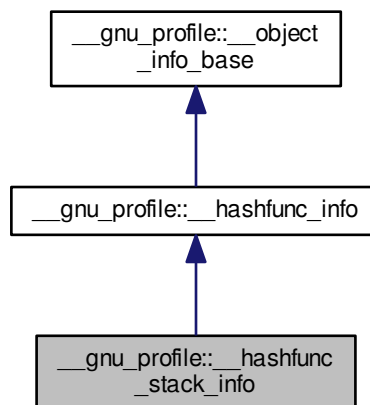
Definition at line 40 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

4.387 `__gnu_profile::__hashfunc_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__hashfunc_stack_info`:



## Public Member Functions

- `__hashfunc_stack_info` (const `__hashfunc_info` &\_\_o)
- `std::string __advice` () const
- void `__destruct` (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__hashfunc_info` &\_\_o)
- `__stack_t __stack` () const
- void `__write` (FILE \*\_\_f) const

## Protected Attributes

- `__stack_t _M_stack`
- bool `_M_valid`

## 4.387.1 Detailed Description

A hash performance instrumentation line in the stack table.

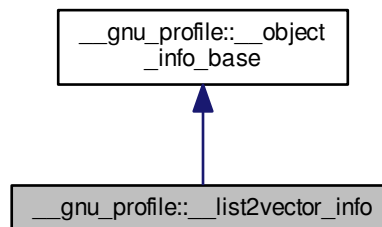
Definition at line 95 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

4.388 `__gnu_profile::__list2vector_info` Class Reference

Inheritance diagram for `__gnu_profile::__list2vector_info`:



## Public Member Functions

- `__list2vector_info` (`__stack_t __stack`)
- `__list2vector_info` (const `__list2vector_info` &\_\_o)
- `std::string __advice` () const

- `bool __is_valid ()`
- `bool __is_valid () const`
- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const \_\_list2vector\_info &__o)`
- `void __opr_insert (std::size_t __shift, std::size_t __size)`
- `void __opr_iterate (std::size_t __num)`
- `std::size_t __resize ()`
- `void __resize (std::size_t __from, std::size_t)`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE * __f) const`

#### Protected Attributes

- `__stack_t M_stack`

##### 4.388.1 Detailed Description

A list-to-vector instrumentation line in the object table.

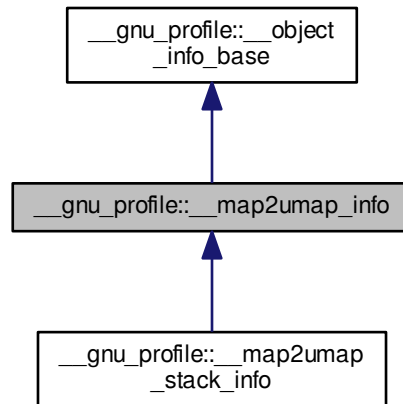
Definition at line 42 of file `profiler_list_to_vector.h`.

The documentation for this class was generated from the following file:

- [profiler\\_list\\_to\\_vector.h](#)

4.389 `__gnu_profile::__map2umap_info` Class Reference

Inheritance diagram for `__gnu_profile::__map2umap_info`:



## Public Member Functions

- `__map2umap_info` (`__stack_t __stack`)
- `__map2umap_info` (`const __map2umap_info &__o`)
- `std::string __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (`const __map2umap_info &__o`)
- `void __record_erase` (`std::size_t __size, std::size_t __count`)
- `void __record_find` (`std::size_t __size`)
- `void __record_insert` (`std::size_t __size, std::size_t __count`)
- `void __record_invalidate` ()
- `void __record_iterate` (`std::size_t __count`)
- `__stack_t __stack` () const
- `void __write` (`FILE *__f`) const

## Protected Attributes

- `__stack_t __M_stack`

## 4.389.1 Detailed Description

A map-to-unordered\_map instrumentation line in the object table.

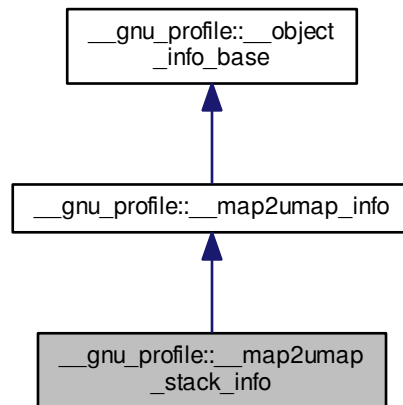
Definition at line 66 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

#### 4.390 `__gnu_profile::__map2umap_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__map2umap_stack_info`:



##### Public Member Functions

- `__map2umap_stack_info` (const [\\_\\_map2umap\\_info](#) &\_\_o)
- `std::string __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const [\\_\\_map2umap\\_info](#) &\_\_o)
- `void __record_erase` (std::size\_t \_\_size, std::size\_t \_\_count)
- `void __record_find` (std::size\_t \_\_size)
- `void __record_insert` (std::size\_t \_\_size, std::size\_t \_\_count)
- `void __record_invalidate` ()
- `void __record_iterate` (std::size\_t \_\_count)
- `__stack_t __stack` () const
- `void __write` (FILE \*\_\_f) const

##### Protected Attributes

- `__stack_t __M_stack`

##### 4.390.1 Detailed Description

A map-to-unordered\_map instrumentation line in the stack table.

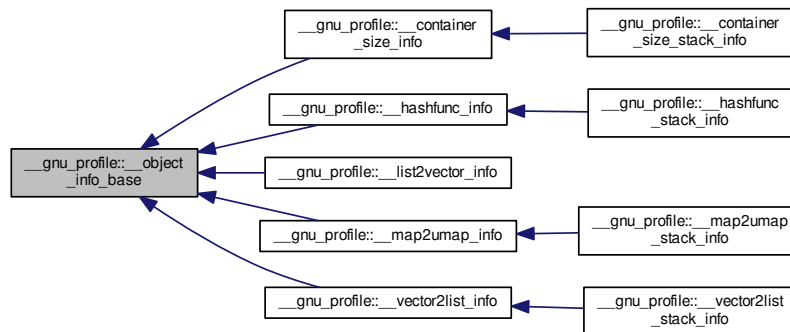
Definition at line 170 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 4.391 \_\_gnu\_profile::\_\_object\_info\_base Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_object\_info\_base:



### Public Member Functions

- **\_\_object\_info\_base** (\_\_stack\_t \_\_stack)
- **\_\_object\_info\_base** (const \_\_object\_info\_base &o)
- bool **is\_valid** () const
- \_\_stack\_t **stack** () const
- virtual void **write** (FILE \*\_\_f) const =0

### Protected Attributes

- \_\_stack\_t **M\_stack**
- bool **M\_valid**

#### 4.391.1 Detailed Description

Base class for a line in the object table.

Definition at line 123 of file profiler\_node.h.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

4.392 `__gnu_profile::__reentrance_guard` Struct Reference

## Static Public Member Functions

- static bool `__get_in` ()
- static bool & `__inside` ()

## 4.392.1 Detailed Description

Reentrance guard.

Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 58 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

4.393 `__gnu_profile::__stack_hash` Class Reference

## Public Member Functions

- `std::size_t operator()` (`__stack_t __s`) const
- bool `operator()` (`__stack_t __stack1`, `__stack_t __stack2`) const

## 4.393.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 89 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

4.394 `__gnu_profile::__stack_info_base< __object_info >` Class Template Reference

## Public Member Functions

- `__stack_info_base` (const `__object_info` & `__info`)=0
- virtual const char \* `__get_id` () const =0
- virtual float `__magnitude` () const =0
- void `__merge` (const `__object_info` & `__info`)=0

## 4.394.1 Detailed Description

```
template<typename __object_info>class __gnu_profile::__stack_info_base< __object_info >
```

Base class for a line in the stack table.

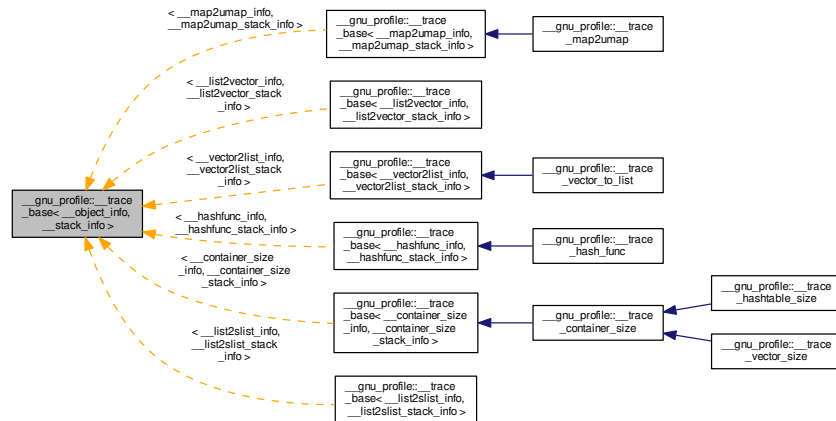
Definition at line 154 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 4.395 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

Inheritance diagram for `__gnu_profile::__trace_base< __object_info, __stack_info >`:



### Public Member Functions

- `void __add_object ( __object_t object, __object_info __info)`
- `void __collect_warnings ( __warning_vector_t &__warnings)`
- `__object_info * __get_object_info ( __object_t __object)`
- `void __retire_object ( __object_t __object)`
- `void __write (FILE * __f)`

### Protected Attributes

- `const char * __id`

#### 4.395.1 Detailed Description

`template<typename __object_info, typename __stack_info>class __gnu_profile::__trace_base< __object_info, __stack_info >`

Base class for all trace producers.

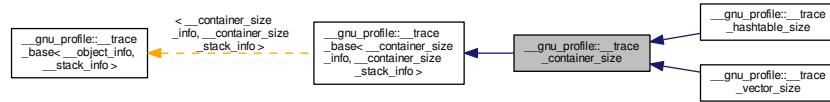
Definition at line 183 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler\\_trace.h](#)

4.396 `__gnu_profile::__trace_container_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_container_size`:



## Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__container_size_info` info)
- void **\_\_collect\_warnings** (`__warning_vector_t` & \_\_warnings)
- void **\_\_construct** (const void \* \_\_obj, `std::size_t` inum)
- void **\_\_destruct** (const void \* \_\_obj, `std::size_t` num, `std::size_t` inum)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_insert** (const `__object_t` \_\_obj, `__stack_t` \_\_stack, `std::size_t` num)
- void **\_\_resize** (const void \* \_\_obj, int from, int to)
- void **\_\_retire\_object** (`__object_t` \_\_object)
- void **\_\_write** (FILE \* \_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.396.1 Detailed Description

Container size instrumentation trace producer.

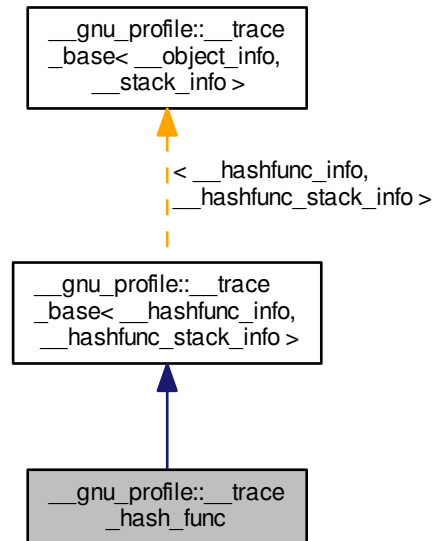
Definition at line 164 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

## 4.397 \_\_gnu\_profile::\_\_trace\_hash\_func Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_hash\_func:



## Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_hashfunc\_info\_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- void **\_\_destruct** (const void \*\_\_obj, std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- [\\_\\_hashfunc\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (\_\_object\_t \_\_obj, \_\_stack\_t \_\_stack)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \*\_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.397.1 Detailed Description

Hash performance instrumentation producer.

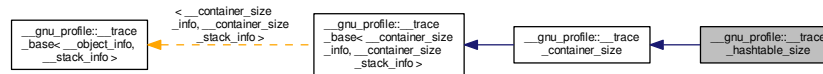
Definition at line 105 of file profiler\_hash\_func.h.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

4.398 `__gnu_profile::__trace_hashtable_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:



## Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__container_size_info` info)
- void **\_\_collect\_warnings** (`__warning_vector_t` &\_\_warnings)
- void **\_\_construct** (const void \*\_\_obj, `std::size_t` \_\_inum)
- void **\_\_destruct** (const void \*\_\_obj, `std::size_t` \_\_num, `std::size_t` \_\_inum)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_insert** (const `__object_t` \_\_obj, `__stack_t` \_\_stack, `std::size_t` \_\_num)
- void **\_\_resize** (const void \*\_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (`__object_t` \_\_object)
- void **\_\_write** (FILE \*\_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.398.1 Detailed Description

Hashtable size instrumentation trace producer.

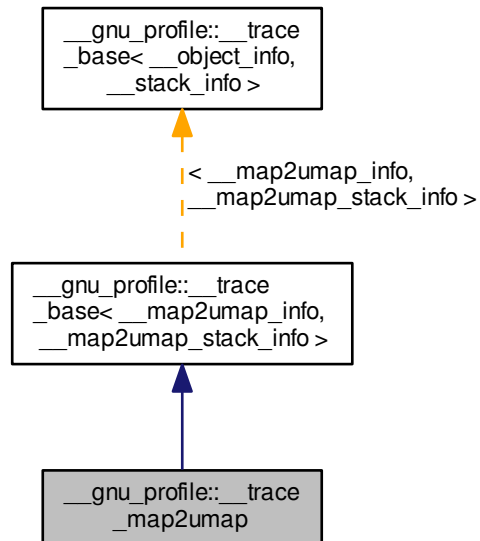
Definition at line 42 of file `profiler_hashtable_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hashtable\\_size.h](#)

## 4.399 \_\_gnu\_profile::\_\_trace\_map2umap Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_map2umap:



## Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_map2umap\_info \_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- [\\_\\_map2umap\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \* \_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.399.1 Detailed Description

Map-to-unordered\_map instrumentation producer.

Definition at line 179 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

4.400 `__gnu_profile::__trace_vector_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



## Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__container_size_info` info)
- void **\_\_collect\_warnings** (`__warning_vector_t` &\_\_warnings)
- void **\_\_construct** (const void \*\_\_obj, `std::size_t` inum)
- void **\_\_destruct** (const void \*\_\_obj, `std::size_t` num, `std::size_t` inum)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_insert** (const `__object_t` \_\_obj, `__stack_t` \_\_stack, `std::size_t` num)
- void **\_\_resize** (const void \*\_\_obj, int from, int to)
- void **\_\_retire\_object** (`__object_t` \_\_object)
- void **\_\_write** (FILE \*\_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.400.1 Detailed Description

Hashtable size instrumentation trace producer.

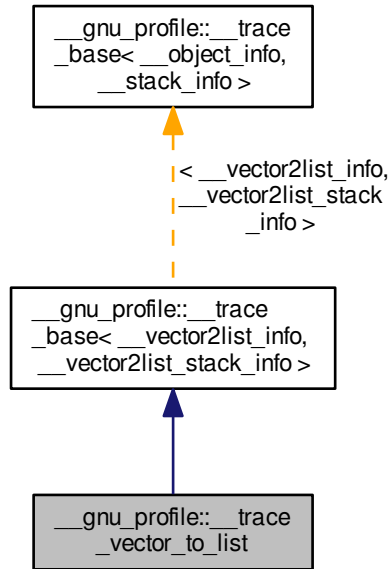
Definition at line 42 of file `profiler_vector_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_size.h](#)

## 4.401 \_\_gnu\_profile::\_\_trace\_vector\_to\_list Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_vector\_to\_list:



## Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_vector2list\_info \_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- void **\_\_destruct** (const void \*\_\_obj)
- [\\_\\_vector2list\\_info](#) \* **\_\_find** (const void \*\_\_obj)
- [\\_\\_vector2list\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (\_\_object\_t \_\_obj, \_\_stack\_t \_\_stack)
- void **\_\_invalid\_operator** (const void \*\_\_obj)
- float **\_\_list\_cost** (std::size\_t \_\_shift, std::size\_t \_\_iterate, std::size\_t \_\_resize)
- void **\_\_opr\_find** (const void \*\_\_obj, std::size\_t \_\_size)
- void **\_\_opr\_insert** (const void \*\_\_obj, std::size\_t \_\_pos, std::size\_t \_\_num)
- void **\_\_opr\_iterate** (const void \*\_\_obj, std::size\_t \_\_num)
- void **\_\_resize** (const void \*\_\_obj, std::size\_t \_\_from, std::size\_t \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- float **\_\_vector\_cost** (std::size\_t \_\_shift, std::size\_t \_\_iterate, std::size\_t \_\_resize)
- void **\_\_write** (FILE \*\_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 4.401.1 Detailed Description

Vector-to-list instrumentation producer.

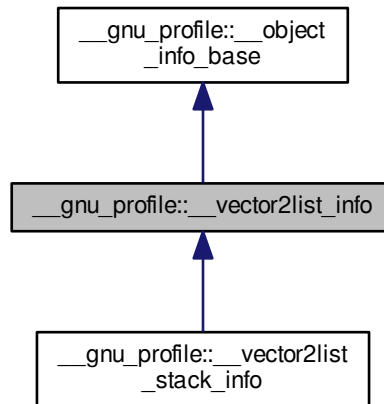
Definition at line 158 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

4.402 `__gnu_profile::__vector2list_info` Class Reference

Inheritance diagram for `__gnu_profile::__vector2list_info`:



## Public Member Functions

- `__vector2list_info` (`__stack_t __stack`)
- `__vector2list_info` (`const __vector2list_info &__o`)
- `std::string __advice` () const
- `bool __is_valid` ()
- `bool __is_valid` () const
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (`const __vector2list_info &__o`)
- `void __opr_find` (`std::size_t __size`)
- `void __opr_insert` (`std::size_t __pos`, `std::size_t __num`)
- `void __opr_iterate` (`std::size_t __num`)
- `std::size_t __resize` ()
- `void __resize` (`std::size_t __from`, `std::size_t`)
- `void __set_invalid` ()

- void `__set_list_cost` (float `__lc`)
- void `__set_vector_cost` (float `__vc`)
- `std::size_t` `__shift_count` ()
- `__stack_t` `__stack` () const
- void `__write` (FILE \*`__f`) const

#### Protected Attributes

- `__stack_t` `_M_stack`

#### 4.402.1 Detailed Description

A vector-to-list instrumentation line in the object table.

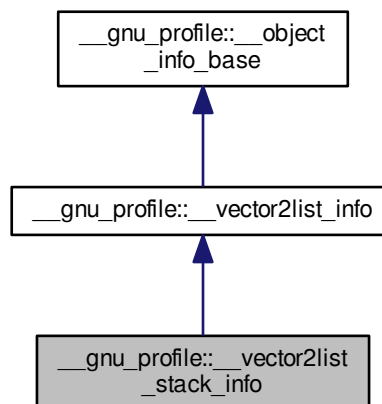
Definition at line 40 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

#### 4.403 `__gnu_profile::__vector2list_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__vector2list_stack_info`:



#### Public Member Functions

- `__vector2list_stack_info` (const [\\_\\_vector2list\\_info](#) &`__o`)
- `std::string` `__advice` () const
- bool `__is_valid` ()
- bool `__is_valid` () const

- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const \_\_vector2list\_info &__o)`
- `void __opr_find (std::size_t __size)`
- `void __opr_insert (std::size_t __pos, std::size_t __num)`
- `void __opr_iterate (std::size_t __num)`
- `std::size_t __resize ()`
- `void __resize (std::size_t __from, std::size_t)`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

#### Protected Attributes

- `__stack_t __M_stack`

#### 4.403.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 148 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 4.404 `__gnu_profile::__warning_data` Struct Reference

#### Public Member Functions

- `__warning_data (float __m, __stack_t __c, const char *__id, const std::string &__msg)`
- `bool operator< (const \_\_warning\_data &__other) const`

#### Public Attributes

- `__stack_t __context`
- `float __magnitude`
- `const char * __warning_id`
- `std::string __warning_message`

#### 4.404.1 Detailed Description

Representation of a warning.

Definition at line 73 of file `profiler_trace.h`.

The documentation for this struct was generated from the following file:

- [profiler\\_trace.h](#)

#### 4.405 `__pool<_Thread>` Class Template Reference

##### 4.405.1 Detailed Description

`template<bool _Thread>class __pool<_Thread>`

Data describing the underlying memory pool, parameterized on threading support.

Definition at line 192 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 4.406 `__reflection_typelist<_Elements>` Struct Template Reference

##### 4.406.1 Detailed Description

`template<typename... _Elements>struct __reflection_typelist<_Elements>`

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.

Definition at line 56 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

#### 4.407 `_Bind<_Signature>` Struct Template Reference

##### 4.407.1 Detailed Description

`template<typename _Signature>struct _Bind<_Signature>`

Type of the function object returned from `bind()`.

Definition at line 1277 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.408 `_Bind_result<_Result, _Signature>` Struct Template Reference

##### 4.408.1 Detailed Description

`template<typename _Result, typename _Signature>struct _Bind_result<_Result, _Signature>`

Type of the function object returned from `bind<R>()`.

Definition at line 1403 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.409 `_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code > Struct Template Reference`

### 4.409.1 Detailed Description

template<typename `_Key`, typename `_Value`, typename `_ExtractKey`, typename `_Equal`, typename `_HashCodeType`, bool `__cache_hash_code`>struct `_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >`

Primary class template `_Equal_helper`.

Definition at line 1156 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.410 `_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference`

### 4.410.1 Detailed Description

template<typename `_Key`, typename `_Value`, typename `_Alloc`, typename `_ExtractKey`, typename `_Equal`, typename `_H1`, typename `_H2`, typename `_Hash`, typename `_RehashPolicy`, typename `_Traits`, bool `_Unique_keys = _Traits::__unique_keys::value`>struct `_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`

Primary class template `_Equality`.

This is for implementing equality comparison for unordered containers, per N3068, by John Lakos and Pablo Halpern. Algorithmically, we follow closely the reference implementations therein.

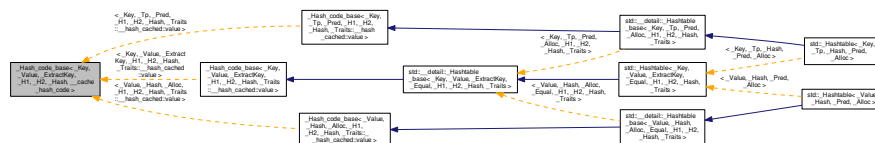
Definition at line 1559 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.411 `_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference`

Inheritance diagram for `_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`:



### 4.411.1 Detailed Description

- hashtable\_policy.h

- hashtable\_policy.h

#### 4.413.1 Detailed Description

```
template<int _Nm, typename _Tp, bool __use_ebo = !_is_final(_Tp) && __is_empty(_Tp)>struct _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >
```

Primary class template `_Hashtable_ebo_helper`.

Helper class using EBO when it is not forbidden, type is not final, and when it worth it, type is empty.

Definition at line 831 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.414 `_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys >` Struct Template Reference

##### 4.414.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Constant_iterators = _Traits::__constant_iterators::value, bool _Unique_keys = _Traits::__unique_keys::value>struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy,
_Traits, _Constant_iterators, _Unique_keys >
```

Primary class template `_Insert`.

Select insert member functions appropriate to `_Hashtable` policy choices.

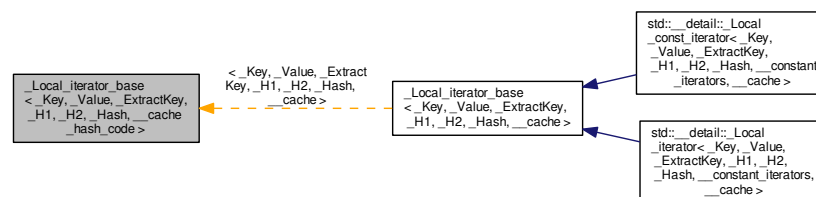
Definition at line 661 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.415 `_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Struct Template Reference

Inheritance diagram for `_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`:



#### 4.415.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code> struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Local_iterator_base`.

Base class for local iterators, used to iterate within a bucket but not between buckets.

Definition at line 882 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- hashtable\_policy.h

#### 4.416 `_Mu<_Arg, _IsBindExp, _IsPlaceholder >` Class Template Reference

#### 4.416.1 Detailed Description

```
template<typename _Arg, bool _IsBindExp = is_bind_expression<_Arg>::value, bool _IsPlaceholder = (is_placeholder<_Arg>::value
> 0)>class _Mu< _Arg, _IsBindExp, _IsPlaceholder >
```

Maps an argument to `bind()` into an actual argument to the bound function object [TR1 3.6.3/5]. Only the first parameter should be specified: the rest are used to determine among the various implementations. Note that, although this class is a function object, it isn't entirely normal because it takes only two parameters regardless of the number of parameters passed to the `bind` expression. The first parameter is the bound argument and the second parameter is a tuple containing references to the rest of the arguments.

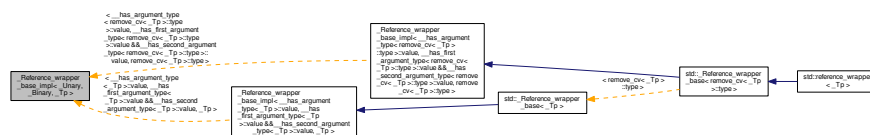
Definition at line 1103 of file functional.

The documentation for this class was generated from the following file:

- functional

#### 4.417 `_Reference_wrapper_base_impl< _Unary, _Binary, _Tp >` Struct Template Reference

Inheritance diagram for `_Reference_wrapper_base_impl<_Unary, _Binary, _Tp>`:



#### 4.417.1 Detailed Description

```
template<bool _Unary, bool _Binary, typename _Tp>struct _Reference_wrapper_base_impl< _Unary, _Binary, _Tp >
```

Knowing which of `unary_function` and `binary_function _Tp` derives from, derives from the same and ensures that `reference_wrapper` will have a weak result type. See cases below.

Definition at line 269 of file functional.

The documentation for this struct was generated from the following file:

- functional

## 4.418 `_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Struct Template Reference

### 4.418.1 Detailed Description

template<typename `_Key`, typename `_Value`, typename `_Alloc`, typename `_ExtractKey`, typename `_Equal`, typename `_H1`, typename `_H2`, typename `_Hash`, typename `_RehashPolicy`, typename `_Traits`>struct `_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Primary class template `_Rehash_base`.

Give hashtable the `max_load_factor` functions and `reserve` iff the rehash policy is `_Prime_rehash_policy`.

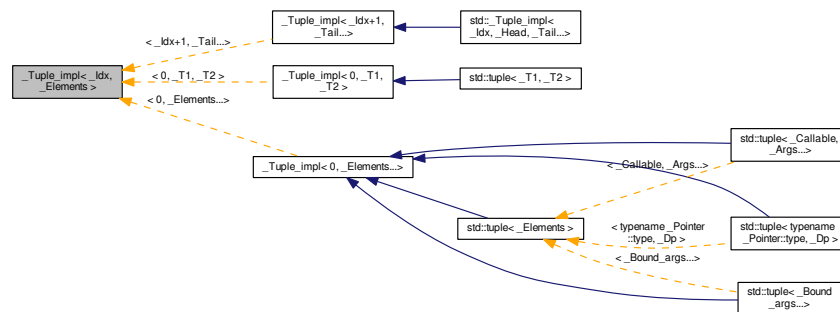
Definition at line 788 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.419 `_Tuple_impl< _Idx, _Elements >` Struct Template Reference

Inheritance diagram for `_Tuple_impl< _Idx, _Elements >`:



### 4.419.1 Detailed Description

template<std::size\_t `_Idx`, typename... `_Elements`>struct `_Tuple_impl< _Idx, _Elements >`

Contains the actual implementation of the `tuple` template, stored as a recursive inheritance hierarchy from the first element (most derived class) to the last (least derived class). The `_Idx` parameter gives the 0-based index of the element stored at this point in the hierarchy; we use it to implement a constant-time `get()` operation.

Definition at line 184 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.420 `common_type< _Tp >` Struct Template Reference

Inherited by `std::common_type< _Tp, _Up, _Vp...>`.

## 4.420.1 Detailed Description

```
template<typename... _Tp>struct common_type< _Tp >
```

`common_type`

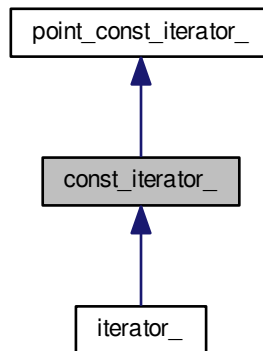
Definition at line 1790 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.421 `const_iterator_` Class Reference

Inheritance diagram for `const_iterator_`:



## Public Types

- `typedef const_pointer_ const\_pointer`
- `typedef const_reference_ const\_reference`
- `typedef _Alloc::difference_type difference\_type`
- `typedef std::forward\_iterator\_tag iterator\_category`
- `typedef pointer_ pointer`
- `typedef reference_ reference`
- `typedef value_type_ value\_type`

## Public Member Functions

- [const\\_iterator\\_](#) ()
- `bool operator!= (const point\_iterator\_ &other) const`
- `bool operator!= (const point\_const\_iterator\_ &other) const`
- `const\_reference operator\* () const`

- `const_iterator_ & operator++ ()`
- `const_iterator_ operator++ (int)`
- `const_pointer operator-> () const`
- `bool operator== (const point_iterator_ &other) const`
- `bool operator== (const point_const_iterator_ &other) const`

#### Protected Types

- `typedef point_const_iterator_ base_type`

#### Protected Member Functions

- `const_iterator_ (const_pointer_ p_value, PB_DS_GEN_POS pos, const PB_DS_CLASS_C_DEC *p_tbl)`

#### Protected Attributes

- `const PB_DS_CLASS_C_DEC * m_p_tbl`
- `const_pointer m_p_value`
- `PB_DS_GEN_POS m_pos`

#### Friends

- `class PB_DS_CLASS_C_DEC`

#### 4.421.1 Detailed Description

Const range-type iterator.

Definition at line 43 of file `unordered_iterator/const_iterator.hpp`.

#### 4.421.2 Member Typedef Documentation

##### 4.421.2.1 `typedef const_pointer_ const_iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 60 of file `unordered_iterator/const_iterator.hpp`.

##### 4.421.2.2 `typedef const_reference_ const_iterator_::const_reference`

Iterator's const reference type.

Definition at line 66 of file `unordered_iterator/const_iterator.hpp`.

##### 4.421.2.3 `typedef _Alloc::difference_type const_iterator_::difference_type`

Difference type.

Definition at line 51 of file `unordered_iterator/const_iterator.hpp`.

4.421.2.4 `typedef std::forward_iterator_tag const_iterator_::iterator_category`

Category.

Definition at line 48 of file `unordered_iterator/const_iterator.hpp`.

4.421.2.5 `typedef pointer_ const_iterator_::pointer`

Iterator's pointer type.

Definition at line 57 of file `unordered_iterator/const_iterator.hpp`.

4.421.2.6 `typedef reference_ const_iterator_::reference`

Iterator's reference type.

Definition at line 63 of file `unordered_iterator/const_iterator.hpp`.

4.421.2.7 `typedef value_type_ const_iterator_::value_type`

Iterator's value type.

Definition at line 54 of file `unordered_iterator/const_iterator.hpp`.

## 4.421.3 Constructor &amp; Destructor Documentation

4.421.3.1 `const_iterator_::const_iterator_( ) [inline]`

Default constructor.

Definition at line 69 of file `unordered_iterator/const_iterator.hpp`.

4.421.3.2 `const_iterator_::const_iterator_( const_pointer_ p_value, PB_DS_GEN_POS pos, const PB_DS_CLASS_C_DEC * p_tbl ) [inline], [protected]`

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a `find()` of a table).

Definition at line 97 of file `unordered_iterator/const_iterator.hpp`.

## 4.421.4 Member Function Documentation

4.421.4.1 `bool point_const_iterator_::operator!=( const point_iterator_ & other ) const [inline], [inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 118 of file `unordered_iterator/point_const_iterator.hpp`.

4.421.4.2 `bool point_const_iterator_::operator!=( const point_const_iterator_ & other ) const [inline], [inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 123 of file `unordered_iterator/point_const_iterator.hpp`.

4.421.4.3 `const_reference point_const_iterator_::operator*( ) const [inline], [inherited]`

Access.

Definition at line 100 of file unordered\_iterator/point\_const\_iterator.hpp.

**4.421.4.4 const\_iterator\_& const\_iterator::operator++ ( ) [inline]**

Increments.

Definition at line 74 of file unordered\_iterator/const\_iterator.hpp.

References `m_p_tbl`.

**4.421.4.5 const\_iterator\_ const\_iterator::operator++ ( int ) [inline]**

Increments.

Definition at line 82 of file unordered\_iterator/const\_iterator.hpp.

References `m_p_tbl`.

**4.421.4.6 const\_pointer point\_const\_iterator::operator-> ( ) const [inline],[inherited]**

Access.

Definition at line 92 of file unordered\_iterator/point\_const\_iterator.hpp.

**4.421.4.7 bool point\_const\_iterator::operator== ( const point\_iterator\_ & other ) const [inline],[inherited]**

Compares content to a different iterator object.

Definition at line 108 of file unordered\_iterator/point\_const\_iterator.hpp.

**4.421.4.8 bool point\_const\_iterator::operator== ( const point\_const\_iterator\_ & other ) const [inline],[inherited]**

Compares content to a different iterator object.

Definition at line 113 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.421.5 Member Data Documentation

**4.421.5.1 const PB\_DS\_CLASS\_C\_DEC\* const\_iterator::m\_p\_tbl [protected]**

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file unordered\_iterator/const\_iterator.hpp.

Referenced by `operator++()`, and `iterator::operator++()`.

The documentation for this class was generated from the following file:

- [unordered\\_iterator/const\\_iterator.hpp](#)

## 4.422 container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_TI > Struct Template Reference

### 4.422.1 Detailed Description

`template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_TI = null_type>struct container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >`

Dispatch mechanism, primary template for associative types.

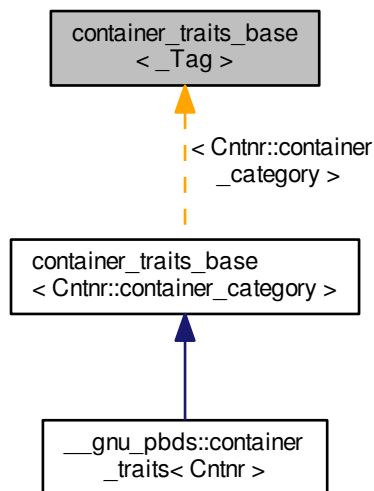
Definition at line 449 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.423 container\_traits\_base< \_Tag > Struct Template Reference

Inheritance diagram for container\_traits\_base< \_Tag >:



##### 4.423.1 Detailed Description

```
template<typename _Tag>struct container_traits_base< _Tag >
```

Primary template, container traits base.

Definition at line 220 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.424 default\_trie\_access\_traits< Key > Struct Template Reference

##### 4.424.1 Detailed Description

```
template<typename Key>struct default_trie_access_traits< Key >
```

Primary template, default\_trie\_access\_traits.

Definition at line 135 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 4.425 `entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>` Struct Template Reference

### 4.425.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>struct entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>
```

Entry compare, primary template.

Definition at line 50 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 4.426 `entry_pred<_VTp, Pred, _Alloc, No_Throw>` Struct Template Reference

### 4.426.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>struct entry_pred<_VTp, Pred, _Alloc, No_Throw>
```

Entry predicate primary class template.

Definition at line 50 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 4.427 `hash<_Tp>` Struct Template Reference

### 4.427.1 Detailed Description

```
template<typename _Tp>struct hash<_Tp>
```

Primary class template hash.

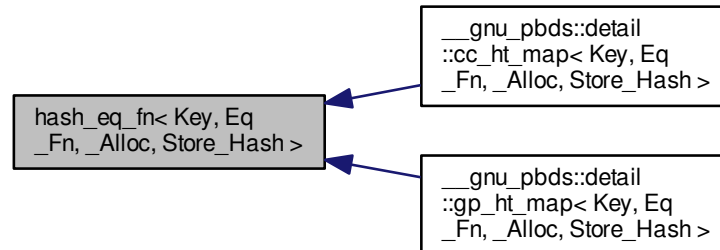
Definition at line 58 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.428 hash\_eq\_fn&lt; Key, Eq\_Fn, \_Alloc, Store\_Hash &gt; Struct Template Reference

Inheritance diagram for hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash >:



## 4.428.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>struct hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >
```

Primary template.

Definition at line 54 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

## 4.429 hash\_load\_check\_resize\_trigger\_size\_base&lt; Size\_Type, Hold\_Size &gt; Class Template Reference

## 4.429.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>class hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

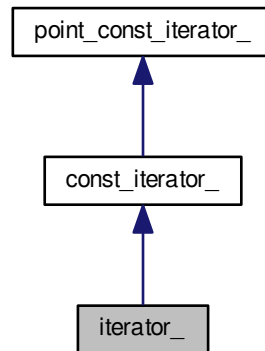
Definition at line 50 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

## 4.430 iterator\_ Class Reference

Inheritance diagram for iterator\_:



## Public Types

- typedef `const_pointer_` [const\\_pointer](#)
- typedef `const_reference_` [const\\_reference](#)
- typedef `_Alloc::difference_type` [difference\\_type](#)
- typedef `std::forward_iterator_tag` [iterator\\_category](#)
- typedef `pointer_` [pointer](#)
- typedef `reference_` [reference](#)
- typedef `value_type_` [value\\_type](#)

## Public Member Functions

- [iterator\\_\(\)](#)
- [operator const point\\_iterator\\_\(\)](#) const
- [operator point\\_iterator\\_\(\)](#)
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [reference operator\\*](#) () const
- [iterator\\_ & operator++](#) ()
- [iterator\\_ operator++](#) (int)
- [pointer operator->](#) () const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

## Protected Types

- typedef [const\\_iterator\\_](#) [base\\_type](#)

### Protected Member Functions

- `iterator_ (pointer p_value, PB_DS_GEN_POS pos, PB_DS_CLASS_C_DEC *p_tbl)`

### Protected Attributes

- `const PB_DS_CLASS_C_DEC * m_p_tbl`
- `const_pointer m_p_value`
- `PB_DS_GEN_POS m_pos`

### Friends

- class `PB_DS_CLASS_C_DEC`

#### 4.430.1 Detailed Description

Range-type iterator.

Definition at line 43 of file iterator.hpp.

#### 4.430.2 Member Typedef Documentation

##### 4.430.2.1 `typedef const_pointer_ iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 60 of file iterator.hpp.

##### 4.430.2.2 `typedef const_reference_ iterator_::const_reference`

Iterator's const reference type.

Definition at line 66 of file iterator.hpp.

##### 4.430.2.3 `typedef _Alloc::difference_type iterator_::difference_type`

Difference type.

Definition at line 51 of file iterator.hpp.

##### 4.430.2.4 `typedef std::forward_iterator_tag iterator_::iterator_category`

Category.

Definition at line 48 of file iterator.hpp.

##### 4.430.2.5 `typedef pointer_ iterator_::pointer`

Iterator's pointer type.

Definition at line 57 of file iterator.hpp.

##### 4.430.2.6 `typedef reference_ iterator_::reference`

Iterator's reference type.

Definition at line 63 of file iterator.hpp.

#### 4.430.2.7 typedef value\_type iterator\_::value\_type

Iterator's value type.

Definition at line 54 of file iterator.hpp.

#### 4.430.3 Constructor & Destructor Documentation

##### 4.430.3.1 iterator\_::iterator\_( ) [inline]

Default constructor.

Definition at line 70 of file iterator.hpp.

##### 4.430.3.2 iterator\_::iterator\_( pointer p\_value, PB\_DS\_GEN\_POS pos, PB\_DS\_CLASS\_C\_DEC \* p\_tbl ) [inline], [protected]

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a find() of a table.

Definition at line 125 of file iterator.hpp.

#### 4.430.4 Member Function Documentation

##### 4.430.4.1 iterator\_::operator const point\_iterator\_( ) const [inline]

Conversion to a point-type iterator.

Definition at line 80 of file iterator.hpp.

##### 4.430.4.2 iterator\_::operator point\_iterator\_( ) [inline]

Conversion to a point-type iterator.

Definition at line 75 of file iterator.hpp.

##### 4.430.4.3 bool point\_const\_iterator\_::operator!=( const point\_iterator\_ & other ) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered\_iterator/point\_const\_iterator.hpp.

##### 4.430.4.4 bool point\_const\_iterator\_::operator!=( const point\_const\_iterator\_ & other ) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered\_iterator/point\_const\_iterator.hpp.

##### 4.430.4.5 reference iterator\_::operator\*( ) const [inline]

Access.

Definition at line 93 of file iterator.hpp.

##### 4.430.4.6 iterator\_ & iterator\_::operator++( ) [inline]

Increments.

Definition at line 101 of file iterator.hpp.

References `const_iterator::m_p_tbl`.

#### 4.430.4.7 iterator\_iterator::operator++ ( int ) [inline]

Increments.

Definition at line 109 of file iterator.hpp.

References `const_iterator::m_p_tbl`.

#### 4.430.4.8 pointer\_iterator::operator-> ( ) const [inline]

Access.

Definition at line 85 of file iterator.hpp.

#### 4.430.4.9 bool point\_const\_iterator::operator== ( const point\_iterator\_ & other ) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 108 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.430.4.10 bool point\_const\_iterator::operator== ( const point\_const\_iterator\_ & other ) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 113 of file unordered\_iterator/point\_const\_iterator.hpp.

### 4.430.5 Member Data Documentation

#### 4.430.5.1 const PB\_DS\_CLASS\_C\_DEC\* const\_iterator::m\_p\_tbl [protected], [inherited]

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file unordered\_iterator/const\_iterator.hpp.

Referenced by `const_iterator::operator++()`, and `operator++()`.

The documentation for this class was generated from the following file:

- [iterator.hpp](#)

## 4.431 owner\_less< \_Tp > Struct Template Reference

### 4.431.1 Detailed Description

```
template<typename _Tp>struct owner_less< _Tp >
```

Primary template `owner_less`.

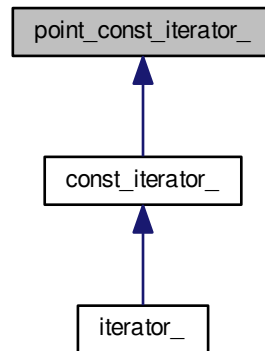
Definition at line 523 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 4.432 point\_const\_iterator\_ Class Reference

Inheritance diagram for point\_const\_iterator\_:



## Public Types

- typedef const\_pointer\_ [const\\_pointer](#)
- typedef const\_reference\_ [const\\_reference](#)
- typedef trivial\_iterator\_difference\_type [difference\\_type](#)
- typedef trivial\_iterator\_tag [iterator\\_category](#)
- typedef pointer\_ [pointer](#)
- typedef reference\_ [reference](#)
- typedef value\_type\_ [value\\_type](#)

## Public Member Functions

- **point\_const\_iterator\_** ([const\\_pointer](#) p\_value)
- [point\\_const\\_iterator\\_](#) ()
- [point\\_const\\_iterator\\_](#) (const [point\\_const\\_iterator\\_](#) &other)
- [point\\_const\\_iterator\\_](#) (const [point\\_iterator\\_](#) &other)
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [const\\_reference](#) [operator\\*](#) () const
- [const\\_pointer](#) [operator->](#) () const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

## Protected Attributes

- [const\\_pointer](#) **m\_p\_value**

## Friends

- class **PB\_DS\_CLASS\_C\_DEC**
- class **point\_iterator\_**

### 4.432.1 Detailed Description

Const point-type iterator.

Definition at line 45 of file unordered\_iterator/point\_const\_iterator.hpp.

### 4.432.2 Member Typedef Documentation

#### 4.432.2.1 typedef const\_pointer\_ point\_const\_iterator\_::const\_pointer

Iterator's const pointer type.

Definition at line 61 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.2.2 typedef const\_reference\_ point\_const\_iterator\_::const\_reference

Iterator's const reference type.

Definition at line 67 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.2.3 typedef trivial\_iterator\_difference\_type point\_const\_iterator\_::difference\_type

Difference type.

Definition at line 52 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.2.4 typedef trivial\_iterator\_tag point\_const\_iterator\_::iterator\_category

Category.

Definition at line 49 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.2.5 typedef pointer\_ point\_const\_iterator\_::pointer

Iterator's pointer type.

Definition at line 58 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.2.6 typedef reference\_ point\_const\_iterator\_::reference

Iterator's reference type.

Definition at line 64 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.2.7 typedef value\_type\_ point\_const\_iterator\_::value\_type

Iterator's value type.

Definition at line 55 of file unordered\_iterator/point\_const\_iterator.hpp.

### 4.432.3 Constructor & Destructor Documentation

#### 4.432.3.1 point\_const\_iterator\_::point\_const\_iterator\_( ) [inline]

Default constructor.

Definition at line 75 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.3.2 point\_const\_iterator\_::point\_const\_iterator\_( const point\_const\_iterator\_ & other ) [inline]

Copy constructor.

Definition at line 80 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.3.3 point\_const\_iterator\_::point\_const\_iterator\_( const point\_iterator\_ & other ) [inline]

Copy constructor.

Definition at line 86 of file unordered\_iterator/point\_const\_iterator.hpp.

### 4.432.4 Member Function Documentation

#### 4.432.4.1 bool point\_const\_iterator\_::operator!=( const point\_iterator\_ & other ) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.4.2 bool point\_const\_iterator\_::operator!=( const point\_const\_iterator\_ & other ) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.4.3 const\_reference point\_const\_iterator\_::operator\*( ) const [inline]

Access.

Definition at line 100 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.4.4 const\_pointer point\_const\_iterator\_::operator->( ) const [inline]

Access.

Definition at line 92 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.4.5 bool point\_const\_iterator\_::operator==( const point\_iterator\_ & other ) const [inline]

Compares content to a different iterator object.

Definition at line 108 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 4.432.4.6 bool point\_const\_iterator\_::operator==( const point\_const\_iterator\_ & other ) const [inline]

Compares content to a different iterator object.

Definition at line 113 of file unordered\_iterator/point\_const\_iterator.hpp.

The documentation for this class was generated from the following file:

- [unordered\\_iterator/point\\_const\\_iterator.hpp](#)

## 4.433 point\_iterator\_ Class Reference

### Public Types

- typedef const\_pointer\_ [const\\_pointer](#)
- typedef const\_reference\_ [const\\_reference](#)
- typedef trivial\_iterator\_difference\_type [difference\\_type](#)
- typedef trivial\_iterator\_tag [iterator\\_category](#)
- typedef pointer\_ [pointer](#)
- typedef reference\_ [reference](#)
- typedef value\_type\_ [value\\_type](#)

### Public Member Functions

- [point\\_iterator\\_](#) ()
- [point\\_iterator\\_](#) (const [point\\_iterator\\_](#) &other)
- **[point\\_iterator\\_](#)** ([pointer](#) p\_value)
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [reference](#) [operator\\*](#) () const
- [pointer](#) [operator->](#) () const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

### Protected Attributes

- [pointer](#) **m\_p\_value**

### Friends

- class **PB\_DS\_CLASS\_C\_DEC**
- class **point\_const\_iterator\_**

#### 4.433.1 Detailed Description

Find type iterator.

Definition at line 43 of file point\_iterator.hpp.

#### 4.433.2 Member Typedef Documentation

##### 4.433.2.1 typedef const\_pointer\_ point\_iterator\_::const\_pointer

Iterator's const pointer type.

Definition at line 59 of file point\_iterator.hpp.

##### 4.433.2.2 typedef const\_reference\_ point\_iterator\_::const\_reference

Iterator's const reference type.

Definition at line 65 of file point\_iterator.hpp.

#### 4.433.2.3 `typedef trivial_iterator_difference_type point_iterator_::difference_type`

Difference type.

Definition at line 50 of file point\_iterator.hpp.

#### 4.433.2.4 `typedef trivial_iterator_tag point_iterator_::iterator_category`

Category.

Definition at line 47 of file point\_iterator.hpp.

#### 4.433.2.5 `typedef pointer_ point_iterator_::pointer`

Iterator's pointer type.

Definition at line 56 of file point\_iterator.hpp.

#### 4.433.2.6 `typedef reference_ point_iterator_::reference`

Iterator's reference type.

Definition at line 62 of file point\_iterator.hpp.

#### 4.433.2.7 `typedef value_type_ point_iterator_::value_type`

Iterator's value type.

Definition at line 53 of file point\_iterator.hpp.

### 4.433.3 Constructor & Destructor Documentation

#### 4.433.3.1 `point_iterator_::point_iterator_( ) [inline]`

Default constructor.

Definition at line 69 of file point\_iterator.hpp.

#### 4.433.3.2 `point_iterator_::point_iterator_( const point_iterator_ & other ) [inline]`

Copy constructor.

Definition at line 75 of file point\_iterator.hpp.

### 4.433.4 Member Function Documentation

#### 4.433.4.1 `bool point_iterator_::operator!=( const point_iterator_ & other ) const [inline]`

Compares content to a different iterator object.

Definition at line 107 of file point\_iterator.hpp.

#### 4.433.4.2 `bool point_iterator_::operator!=( const point_const_iterator_ & other ) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 112 of file point\_iterator.hpp.

4.433.4.3 reference point\_iterator::operator\*( ) const [inline]

Access.

Definition at line 89 of file point\_iterator.hpp.

4.433.4.4 pointer point\_iterator::operator->( ) const [inline]

Access.

Definition at line 81 of file point\_iterator.hpp.

4.433.4.5 bool point\_iterator::operator==( const point\_iterator\_ & other ) const [inline]

Compares content to a different iterator object.

Definition at line 97 of file point\_iterator.hpp.

4.433.4.6 bool point\_iterator::operator==( const point\_const\_iterator\_ & other ) const [inline]

Compares content to a different iterator object.

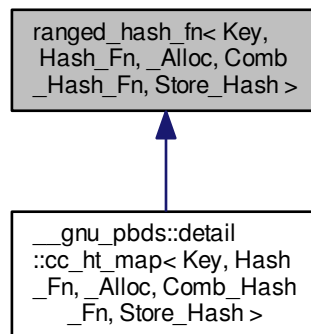
Definition at line 102 of file point\_iterator.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterator.hpp](#)

## 4.434 ranged\_hash\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, Store\_Hash > Class Template Reference

Inheritance diagram for ranged\_hash\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, Store\_Hash >:



### 4.434.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>class ranged_hash_fn<
Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >
```

Primary template.

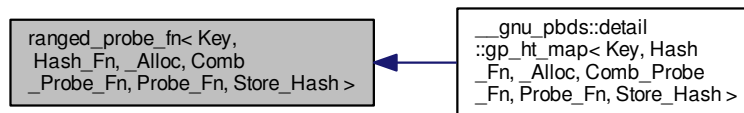
Definition at line 55 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.435 `ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`:



##### 4.435.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_
Hash>class ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >
```

Primary template.

Definition at line 55 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

#### 4.436 `result_of< _Signature >` Class Template Reference

##### 4.436.1 Detailed Description

```
template<typename _Signature>class result_of< _Signature >
```

`result_of`

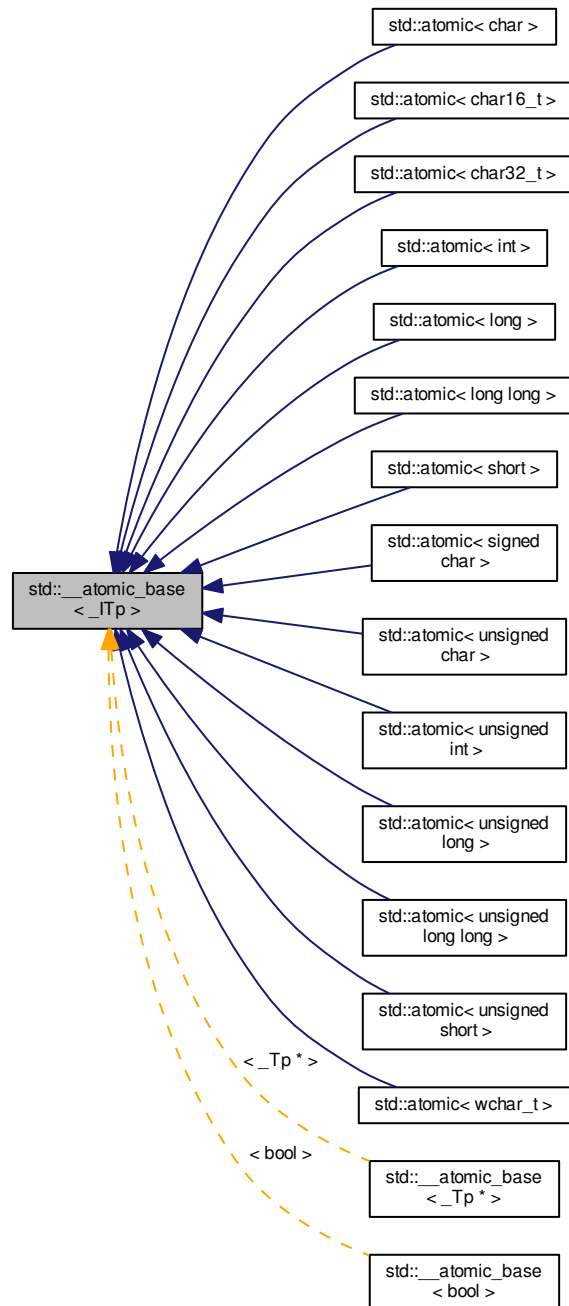
Definition at line 1878 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 4.437 std::\_\_atomic\_base&lt;\_ITp&gt; Struct Template Reference

Inheritance diagram for std::\_\_atomic\_base<\_ITp>:



## Public Member Functions

- **\_\_atomic\_base** (const [\\_\\_atomic\\_base](#) &)=delete
- constexpr **\_\_atomic\_base** ([\\_\\_int\\_type](#) \_\_i) noexcept
- bool **compare\_exchange\_strong** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** ([\\_\\_int\\_type](#) &\_\_i1, [\\_\\_int\\_type](#) \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [\\_\\_int\\_type](#) **exchange** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- [\\_\\_int\\_type](#) **exchange** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [\\_\\_int\\_type](#) **fetch\_add** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- [\\_\\_int\\_type](#) **fetch\_add** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [\\_\\_int\\_type](#) **fetch\_and** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- [\\_\\_int\\_type](#) **fetch\_and** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [\\_\\_int\\_type](#) **fetch\_or** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- [\\_\\_int\\_type](#) **fetch\_or** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [\\_\\_int\\_type](#) **fetch\_sub** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- [\\_\\_int\\_type](#) **fetch\_sub** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [\\_\\_int\\_type](#) **fetch\_xor** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- [\\_\\_int\\_type](#) **fetch\_xor** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- [\\_\\_int\\_type](#) **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- [\\_\\_int\\_type](#) **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** [\\_\\_int\\_type](#) () const noexcept
- **operator** [\\_\\_int\\_type](#) () const volatile noexcept
- [\\_\\_int\\_type](#) **operator&=** ([\\_\\_int\\_type](#) \_\_i) noexcept
- [\\_\\_int\\_type](#) **operator&=** ([\\_\\_int\\_type](#) \_\_i) volatile noexcept
- [\\_\\_int\\_type](#) **operator++** (int) noexcept
- [\\_\\_int\\_type](#) **operator++** (int) volatile noexcept
- [\\_\\_int\\_type](#) **operator++** () noexcept
- [\\_\\_int\\_type](#) **operator++** () volatile noexcept
- [\\_\\_int\\_type](#) **operator+=** ([\\_\\_int\\_type](#) \_\_i) noexcept
- [\\_\\_int\\_type](#) **operator+=** ([\\_\\_int\\_type](#) \_\_i) volatile noexcept
- [\\_\\_int\\_type](#) **operator--** (int) noexcept
- [\\_\\_int\\_type](#) **operator--** (int) volatile noexcept
- [\\_\\_int\\_type](#) **operator--** () noexcept
- [\\_\\_int\\_type](#) **operator--** () volatile noexcept
- [\\_\\_int\\_type](#) **operator-=** ([\\_\\_int\\_type](#) \_\_i) noexcept

- `__int_type operator= (__int_type __i) volatile noexcept`
- `__atomic_base & operator= (const __atomic_base &)=delete`
- `__atomic_base & operator= (const __atomic_base &) volatile=delete`
- `__int_type operator= (__int_type __i) noexcept`
- `__int_type operator= (__int_type __i) volatile noexcept`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.437.1 Detailed Description

template<typename \_ITp>struct std::\_\_atomic\_base< \_ITp >

Base class for atomic integrals.

Definition at line 349 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

#### 4.438 std::\_\_atomic\_base<\_PTp \* > Struct Template Reference

##### Public Member Functions

- `__atomic_base (const __atomic_base &)=delete`
- `constexpr __atomic_base (__pointer_type __p) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const volatile noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator __pointer_type () const noexcept`
- `operator __pointer_type () const volatile noexcept`
- `__pointer_type operator++ (int) noexcept`
- `__pointer_type operator++ (int) volatile noexcept`
- `__pointer_type operator++ () noexcept`
- `__pointer_type operator++ () volatile noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) noexcept`

- `__pointer_type operator+= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator-- (int) noexcept`
- `__pointer_type operator-- (int) volatile noexcept`
- `__pointer_type operator-- () noexcept`
- `__pointer_type operator-- () volatile noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) volatile noexcept`
- `__atomic_base & operator= (const __atomic_base &) delete`
- `__atomic_base & operator= (const __atomic_base &) volatile delete`
- `__pointer_type operator= (__pointer_type __p) noexcept`
- `__pointer_type operator= (__pointer_type __p) volatile noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.438.1 Detailed Description

```
template<typename _PTp>struct std::__atomic_base< _PTp * >
```

Partial specialization for pointer types.

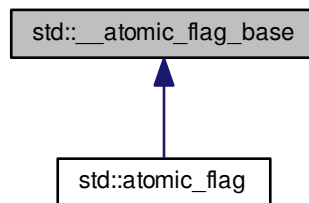
Definition at line 665 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

#### 4.439 std::\_\_atomic\_flag\_base Struct Reference

Inheritance diagram for `std::__atomic_flag_base`:



#### Public Attributes

- `__atomic_flag_data_type _M_i`

## 4.439.1 Detailed Description

Base type for atomic\_flag.

Base type is POD with data, allowing atomic\_flag to derive from it and meet the standard layout type requirement. In addition to compatibility with a C interface, this allows different implementations of atomic\_flag to use the same atomic operation functions, via a standard conversion to the \_\_atomic\_flag\_base argument.

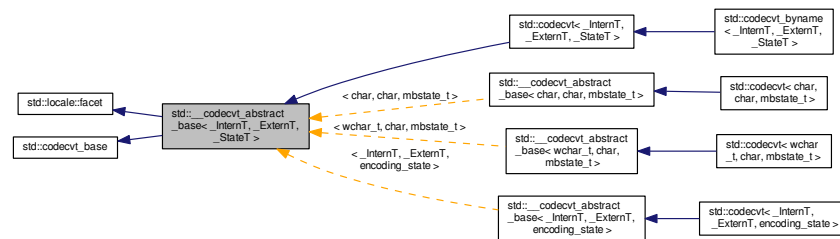
Definition at line 261 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 4.440 std::\_\_codecvt\_abstract\_base&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Protected Member Functions

- **\_\_codecvt\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* & \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* & \_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const =0 throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* & \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* & \_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* & \_\_to\_next) const =0

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## 4.440.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT
>
```

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 68 of file codecvt.h.

## 4.440.2 Member Function Documentation

4.440.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [protected], [pure virtual]`

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See Also

out for more information.

Implemented in [std::codecvt< wchar\\_t, char, mbstate\\_t >](#), [std::codecvt< char, char, mbstate\\_t >](#), [std::codecvt< \\_InternT, \\_ExternT, \\_StateT >](#), and [std::codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#).

Referenced by [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >::out\(\)](#).

**4.440.2.2** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const [inline]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

**4.440.2.3** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [inline]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

4.440.2.4 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` `[inline]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

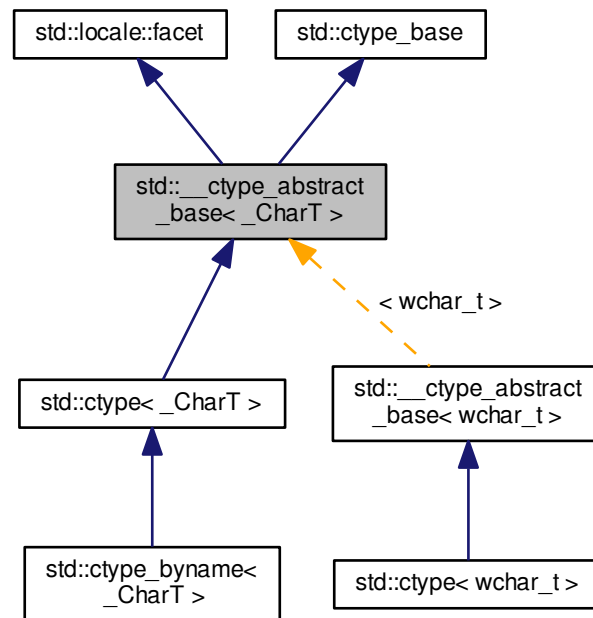
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.441 std::\_\_ctype\_abstract\_base&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::\_\_ctype\_abstract\_base<\_CharT>:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef `_CharT` **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- bool **is** (mask `__m`, **char\_type** `__c`) const
- const **char\_type** \* **is** (const **char\_type** \* `__lo`, const **char\_type** \* `__hi`, mask \* `__vec`) const
- char **narrow** (**char\_type** `__c`, char `__default`) const
- const **char\_type** \* **narrow** (const **char\_type** \* `__lo`, const **char\_type** \* `__hi`, char `__default`, char \* `__to`) const
- const **char\_type** \* **scan\_is** (mask `__m`, const **char\_type** \* `__lo`, const **char\_type** \* `__hi`) const
- const **char\_type** \* **scan\_not** (mask `__m`, const **char\_type** \* `__lo`, const **char\_type** \* `__hi`) const
- **char\_type** **tolower** (**char\_type** `__c`) const
- const **char\_type** \* **tolower** (**char\_type** \* `__lo`, const **char\_type** \* `__hi`) const
- **char\_type** **toupper** (**char\_type** `__c`) const
- const **char\_type** \* **toupper** (**char\_type** \* `__lo`, const **char\_type** \* `__hi`) const
- **char\_type** **widen** (char `__c`) const
- const char \* **widen** (const char \* `__lo`, const char \* `__hi`, **char\_type** \* `__to`) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- **\_\_ctype\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const =0
- virtual const char\_type \* **do\_is** (const char\_type \*\_\_lo, const char\_type \*\_\_hi, mask \*\_\_vec) const =0
- virtual char **do\_narrow** (char\_type \_\_c, char \_\_dfault) const =0
- virtual const char\_type \* **do\_narrow** (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_to) const =0
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const =0
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const =0
- virtual char\_type **do\_tolower** (char\_type \_\_c) const =0
- virtual const char\_type \* **do\_tolower** (char\_type \*\_\_lo, const char\_type \*\_\_hi) const =0
- virtual char\_type **do\_toupper** (char\_type \_\_c) const =0
- virtual const char\_type \* **do\_toupper** (char\_type \*\_\_lo, const char\_type \*\_\_hi) const =0
- virtual char\_type **do\_widen** (char \_\_c) const =0
- virtual const char \* **do\_widen** (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const =0

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.441.1 Detailed Description

template<typename \_CharT>class std::\_\_ctype\_abstract\_base<\_CharT>

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file locale\_facets.h.

## 4.441.2 Member Typedef Documentation

## 4.441.2.1 template&lt;typename \_CharT&gt; typedef \_CharT std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::char\_type

Typedef for the template parameter.

Definition at line 148 of file locale\_facets.h.

## 4.441.3 Member Function Documentation

## 4.441.3.1 template&lt;typename \_CharT&gt; virtual bool std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::do\_is( mask \_\_m, char\_type \_\_c ) const [protected], [pure virtual]

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

## Returns

(M & \_\_m) != 0.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::is\(\)](#).

## 4.441.3.2 template&lt;typename \_CharT&gt; virtual const char\_type\* std::\_\_ctype\_abstract\_base&lt;\_CharT&gt;::do\_is( const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec ) const [protected], [pure virtual]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**`__hi`.Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

4.441.3.3 `template<typename _CharT> virtual char std::__ctype_abstract_base<_CharT>::do_narrow ( char_type __c, char __dfault ) const` `[protected]`, `[pure virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

**Returns**The converted `char`.Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).Referenced by `std::__ctype_abstract_base<wchar_t>::narrow()`.

4.441.3.4 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` `[protected]`, `[pure virtual]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**`__hi`.Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

4.441.3.5 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi ) const` [protected],[pure virtual]

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching char\_type if found, else `__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_is()`.

4.441.3.6 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi ) const` [protected],[pure virtual]

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching char\_type if found, else `__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_not()`.

4.441.3.7 `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_tolower (char_type __c ) const` [protected],[pure virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

## Returns

The lowercase char\_type if convertible, else `__c`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::tolower()`.

**4.441.3.8** `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_tolower (char_type * __lo, const char_type * __hi) const` `[protected]`, `[pure virtual]`

Convert array to lowercase.

This virtual function converts each char\_type in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

**4.441.3.9** `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_toupper (char_type __c) const` `[protected]`, `[pure virtual]`

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

## Returns

The uppercase char\_type if convertible, else `__c`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::toupper()`.

4.441.3.10 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_toupper ( char_type * __lo, const char_type * __hi ) const` [protected], [pure virtual]

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

4.441.3.11 `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_widen ( char __c ) const` [protected], [pure virtual]

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted `char_type`

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::widen()`.

4.441.3.12 `template<typename _CharT> virtual const char* std::__ctype_abstract_base<_CharT>::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` [protected], [pure virtual]

Widen char array.

This function converts each `char` in the input to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start range.           |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

4.441.3.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c ) const [inline]`

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__c</code> | The <code>char_type</code> to compare the mask of. |
| <code>__m</code> | The mask to compare against.                       |

## Returns

`(M & __m) != 0`.

Definition at line 162 of file `locale_facets.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

4.441.3.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [inline]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

## Returns

`__hi`.

Definition at line 179 of file `locale_facets.h`.

4.441.3.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow ( char_type __c, char __dfault ) const [inline]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

#### Returns

The converted char.

Definition at line 324 of file `locale_facets.h`.

Referenced by `std::time_put<_CharT, _Outlter>::put()`.

**4.441.3.16** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline]`

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 346 of file `locale_facets.h`.

**4.441.3.17** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline]`

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else \_\_hi.

Definition at line 195 of file locale\_facets.h.

**4.441.3.18** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else \_\_hi.

Definition at line 211 of file locale\_facets.h.

**4.441.3.19** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const [inline]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Definition at line 254 of file locale\_facets.h.

**4.441.3.20** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower ( char_type * __lo, const char_type * __hi ) const [inline]`

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi.`

Definition at line 269 of file locale\_facets.h.

4.441.3.21 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

## Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file locale\_facets.h.

4.441.3.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi.`

Definition at line 240 of file locale\_facets.h.

4.441.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline]`

Widen char to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

## Returns

The converted char\_type.

Definition at line 286 of file locale\_facets.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::money\_put< \_CharT, \_OutIter >::do\_put(), std::tr2::operator<<(), and std::operator<<().

4.441.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base< _CharT >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codecvt conversions. See codecvt for that.

## Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 305 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.442 std::\_\_debug::bitset&lt; \_Nb &gt; Class Template Reference

Inherits `bitset< _Nb >`.

## Public Types

- typedef `_Base::reference` **reference**

## Public Member Functions

- constexpr **bitset** (unsigned long long \_\_val) noexcept
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
**bitset** (const std::basic\_string< \_CharT, \_Traits, \_Alloc > &\_\_str, typename std::basic\_string< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, typename std::basic\_string< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n=(std::basic\_string< \_CharT, \_Traits, \_Alloc >::npos))
- template<class \_CharT, class \_Traits, class \_Alloc >  
**bitset** (const std::basic\_string< \_CharT, \_Traits, \_Alloc > &\_\_str, typename std::basic\_string< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename std::basic\_string< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- **bitset** (const [\\_Base](#) &\_\_x)

- template<typename \_CharT >  
**bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- [bitset](#)< \_Nb > & [flip](#) () noexcept
- [bitset](#)< \_Nb > & [flip](#) (size\_t \_\_pos)
- bool [operator!=](#) (const [bitset](#)< \_Nb > &\_\_rhs) const noexcept
- [bitset](#)< \_Nb > & [operator&=](#) (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > [operator<<](#) (size\_t \_\_pos) const noexcept
- [bitset](#)< \_Nb > & [operator<<=](#) (size\_t \_\_pos) noexcept
- bool [operator==](#) (const [bitset](#)< \_Nb > &\_\_rhs) const noexcept
- [bitset](#)< \_Nb > [operator>>](#) (size\_t \_\_pos) const noexcept
- [bitset](#)< \_Nb > & [operator>>=](#) (size\_t \_\_pos) noexcept
- reference [operator\[\]](#) (size\_t \_\_pos)
- constexpr bool [operator\[\]](#) (size\_t \_\_pos) const
- [bitset](#)< \_Nb > & [operator^=](#) (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > & [operator|=](#) (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > [operator~](#) () const noexcept
- [bitset](#)< \_Nb > & [reset](#) () noexcept
- [bitset](#)< \_Nb > & [reset](#) (size\_t \_\_pos)
- [bitset](#)< \_Nb > & [set](#) () noexcept
- [bitset](#)< \_Nb > & [set](#) (size\_t \_\_pos, bool \_\_val=true)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) () const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)  
< \_CharT > > [to\\_string](#) () const
- template<class \_CharT, class \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)  
< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<class \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) () const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) (char \_\_zero, char \_\_one= '1') const

## 4.442.1 Detailed Description

```
template<size_t _Nb>class std::__debug::bitset< _Nb >
```

Class std::bitset with additional safety/checking/debug instrumentation.

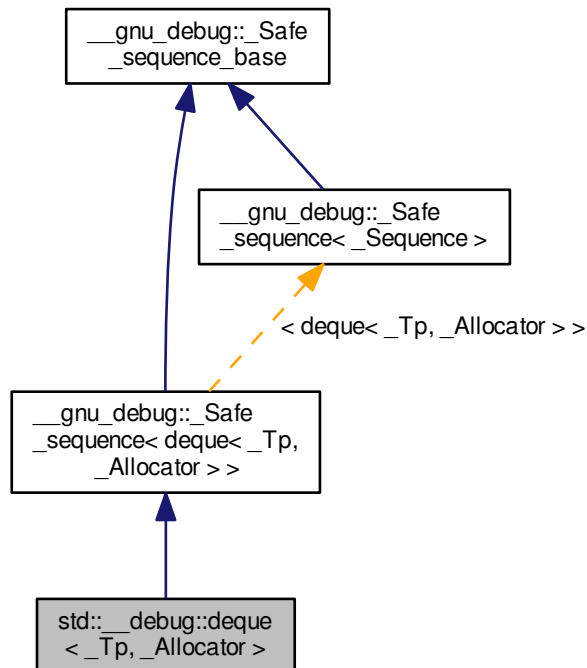
Definition at line 42 of file debug/bitset.

The documentation for this class was generated from the following file:

- [debug/bitset](#)

## 4.443 std::\_\_debug::deque&lt; \_Tp, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::deque< \_Tp, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, deque >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**

- typedef std::reverse\_iterator  
< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef  
\_\_gnu\_debug:: Safe\_iterator  
< \_Base\_iterator, deque > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef std::reverse\_iterator  
< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **deque** (const \_Allocator &\_\_a= \_Allocator())
- **deque** (size\_type \_\_n)
- **deque** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a= \_Allocator())
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>  
**deque** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a= \_Allocator())
- **deque** (const deque &\_\_x)
- **deque** (const \_Base &\_\_x)
- **deque** (deque &&\_\_x)
- **deque** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- void **\_M\_attach** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- **\_Base & \_M\_base** () noexcept
- const **\_Base & \_M\_base** () const noexcept
- void **\_M\_detach** (\_Safe\_iterator\_base \*\_\_it)
- void **\_M\_detach\_single** (\_Safe\_iterator\_base \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_t)
- void **assign** (initializer\_list< value\_type > \_\_l)
- reference **back** ()
- const\_reference **back** () const
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- **const\_reverse\_iterator crbegin** () const noexcept
- **const\_reverse\_iterator crend** () const noexcept
- template<typename... \_Args>  
**iterator emplace** (iterator \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void **emplace\_back** (\_Args &&...\_\_args)

- template<typename... \_Args>  
void **emplace\_front** (\_Args &&...\_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **iterator erase** (**iterator** \_\_position)
- **iterator erase** (**iterator** \_\_first, **iterator** \_\_last)
- reference **front** ()
- const\_reference **front** () const
- **iterator insert** (**iterator** \_\_position, const \_Tp &\_\_x)
- **iterator insert** (**iterator** \_\_position, \_Tp &&\_\_x)
- void **insert** (**iterator** \_\_p, **initializer\_list**< value\_type > \_\_l)
- void **insert** (**iterator** \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<class \_InputIterator, typename = std::\_\_RequireInputIter<\_InputIterator>>>  
void **insert** (**iterator** \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- **deque & operator=** (const **deque** &\_\_x)
- **deque & operator=** (**deque** &&\_\_x)
- **deque & operator=** (**initializer\_list**< value\_type > \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (const \_Tp &\_\_x)
- void **push\_back** (\_Tp &&\_\_x)
- void **push\_front** (const \_Tp &\_\_x)
- void **push\_front** (\_Tp &&\_\_x)
- **reverse\_iterator rbegin** () noexcept
- **const\_reverse\_iterator rbegin** () const noexcept
- **reverse\_iterator rend** () noexcept
- **const\_reverse\_iterator rend** () const noexcept
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- void **shrink\_to\_fit** ()
- void **swap** (**deque** &\_\_x)

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- unsigned int **\_M\_version**

#### Protected Member Functions

- void **\_M\_detach\_all** ()
- void **\_M\_detach\_singular** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () throw ()
- void **\_M\_revalidate\_singular** ()
- void **\_M\_swap** (\_Safe\_sequence\_base &\_\_x)

## 4.443.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__debug::deque<_Tp, _Allocator>
```

Class std::deque with safety/checking/debug instrumentation.

Definition at line 42 of file debug/deque.

## 4.443.2 Member Function Documentation

4.443.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.443.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.443.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.443.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.443.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.443.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected], [inherited]

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.443.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.443.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.443.2.9 void \_\_gnu\_debug::Safe\_sequence< deque< \_Tp, \_Allocator > >::M\_invalidate\_if ( \_Predicate \_\_pred )  
[inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which \_\_pred(*x*) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.443.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.443.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.443.2.12 void \_\_gnu\_debug::Safe\_sequence< deque< \_Tp, \_Allocator > >::M\_transfer\_from\_if ( \_Safe\_sequence< deque< \_Tp, \_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which \_\_pred(*x*) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

#### 4.443.3 Member Data Documentation

4.443.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

4.443.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

4.443.3.3 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe\_base.h.

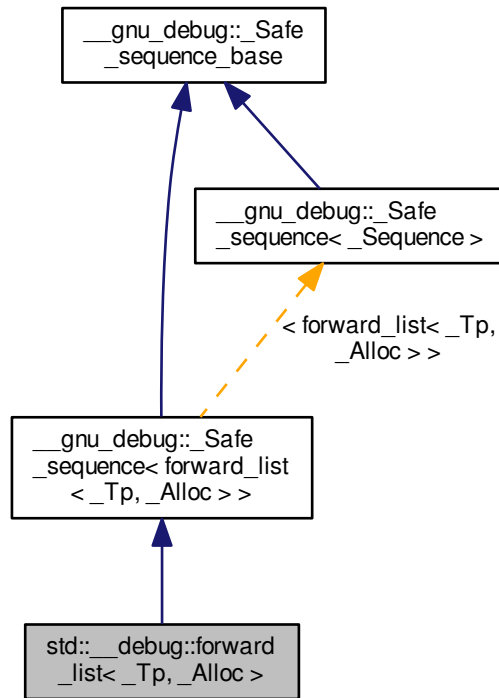
Referenced by \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

The documentation for this class was generated from the following file:

- [debug/deque](#)

## 4.444 std::\_\_debug::forward\_list&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::forward\_list< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, forward_list >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, forward_list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **forward\_list** (const \_Alloc &\_\_al=\_Alloc())
- **forward\_list** (const [forward\\_list](#) &\_\_list, const \_Alloc &\_\_al)
- **forward\_list** ([forward\\_list](#) &&\_\_list, const \_Alloc &\_\_al)
- **forward\_list** (size\_type \_\_n, const \_Alloc &\_\_al=\_Alloc())
- **forward\_list** (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al=\_Alloc())
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
**forward\_list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Alloc &\_\_al=\_Alloc())
- **forward\_list** (const [forward\\_list](#) &\_\_list)
- **forward\_list** ([forward\\_list](#) &&\_\_list) noexcept
- **forward\_list** ([std::initializer\\_list](#)<\_Tp> \_\_il, const \_Alloc &\_\_al=\_Alloc())
- void **\_M\_attach** (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- **\_Base** & **\_M\_base** () noexcept
- const **\_Base** & **\_M\_base** () const noexcept
- void **\_M\_detach** (\_Safe\_iterator\_base \* \_\_it)
- void **\_M\_detach\_single** (\_Safe\_iterator\_base \* \_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence & \_\_from, \_Predicate \_\_pred)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_val)
- void **assign** ([std::initializer\\_list](#)<\_Tp> \_\_il)
- **iterator before\_begin** () noexcept
- **const\_iterator before\_begin** () const noexcept
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbefore\_begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- template<typename... \_Args>  
**iterator emplace\_after** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **iterator erase\_after** (const\_iterator \_\_pos)
- **iterator erase\_after** (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- **iterator insert\_after** (const\_iterator \_\_pos, const \_Tp &\_\_val)
- **iterator insert\_after** (const\_iterator \_\_pos, \_Tp &&\_\_val)
- **iterator insert\_after** (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
**iterator insert\_after** (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- **iterator insert\_after** (const\_iterator \_\_pos, [std::initializer\\_list](#)<\_Tp> \_\_il)
- void **merge** ([forward\\_list](#) &&\_\_list)
- void **merge** ([forward\\_list](#) &\_\_list)
- template<typename \_Comp>  
void **merge** ([forward\\_list](#) &&\_\_list, \_Comp \_\_comp)

- template<typename \_Comp >  
void **merge** (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & **operator=** (const forward\_list &\_\_list)
- forward\_list & **operator=** (forward\_list &&\_\_list) noexcept(\_Node\_alloc\_traits
- forward\_list & **operator=** (std::initializer\_list< \_Tp > \_\_il)
- void **pop\_front** ()
- void **remove** (const \_Tp &\_\_val)
- template<typename \_Pred >  
void **remove\_if** (\_Pred \_\_pred)
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const value\_type &\_\_val)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **swap** (forward\_list &\_\_list) noexcept(\_Node\_alloc\_traits
- void **unique** ()
- template<typename \_BinPred >  
void **unique** (\_BinPred \_\_binary\_pred)

#### Public Attributes

- \_Safe\_iterator\_base \* \_M\_const\_iterators
- \_Safe\_iterator\_base \* \_M\_iterators
- unsigned int \_M\_version

#### Protected Member Functions

- void \_M\_detach\_all ()
- void \_M\_detach\_singular ()
- \_\_gnu\_cxx::\_\_mutex & \_M\_get\_mutex () throw ()
- void \_M\_revalidate\_singular ()
- void \_M\_swap (\_Safe\_sequence\_base &\_\_x)

#### 4.444.1 Detailed Description

template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>>class std::\_\_debug::forward\_list< \_Tp, \_Alloc >

Class std::forward\_list with safety/checking/debug instrumentation.

Definition at line 44 of file debug/forward\_list.

#### 4.444.2 Member Function Documentation

4.444.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.444.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.444.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.444.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.444.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.444.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.444.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.444.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.444.2.9 void \_\_gnu\_debug::Safe\_sequence<forward\_list<\_Tp, \_Alloc>>::M\_invalidate\_if ( \_Predicate \_\_pred )  
[inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.444.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.444.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.444.2.12 void \_\_gnu\_debug::\_Safe\_sequence< forward\_list<\_Tp, \_Alloc> >::M\_transfer\_from\_if (   
 \_Safe\_sequence< forward\_list<\_Tp, \_Alloc> > & \_\_from, Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### 4.444.3 Member Data Documentation

4.444.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.444.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.444.3.3 unsigned int \_\_gnu\_debug::\_Safe\_sequence\_base::M\_version [mutable],[inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

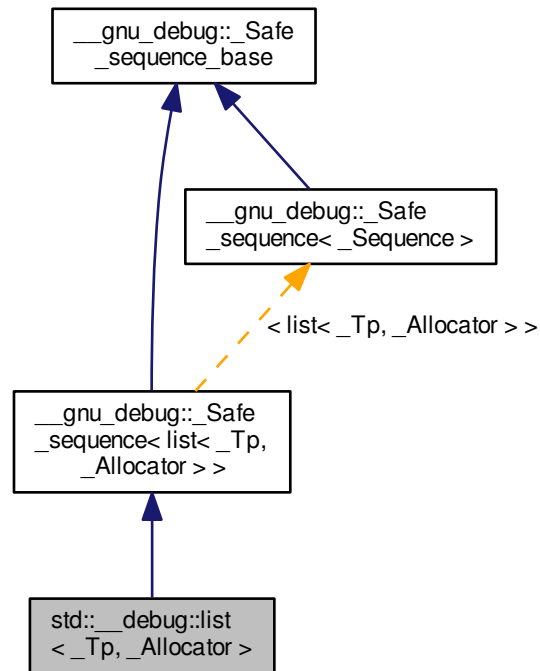
Referenced by `__gnu_debug::_Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

## 4.445 std::\_\_debug::list&lt; \_Tp, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::list< \_Tp, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, list >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **list** (const \_Allocator &\_\_a=\_Allocator())
- **list** (size\_type \_\_n)
- **list** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a=\_Allocator())
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
  **list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a=\_Allocator())
- **list** (const **list** &\_\_x)
- **list** (const **\_Base** &\_\_x)
- **list** (**list** &&\_\_x) noexcept
- **list** (**initializer\_list**< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- void **\_M\_attach** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- **\_Base** & **\_M\_base** () noexcept
- const **\_Base** & **\_M\_base** () const noexcept
- void **\_M\_detach** (\_Safe\_iterator\_base \*\_\_it)
- void **\_M\_detach\_single** (\_Safe\_iterator\_base \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- void **assign** (**initializer\_list**< value\_type > \_\_l)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
  void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_t)
- reference **back** ()
- const\_reference **back** () const
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- **const\_reverse\_iterator crbegin** () const noexcept
- **const\_reverse\_iterator crend** () const noexcept
- template<typename... \_Args>  
  **iterator emplace** (**iterator** \_\_position, \_Args &&... \_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **iterator erase** (**iterator** \_\_position)
- **iterator erase** (**iterator** \_\_position, **iterator** \_\_last)
- reference **front** ()
- const\_reference **front** () const
- **iterator insert** (**iterator** \_\_position, const \_Tp &\_\_x)
- **iterator insert** (**iterator** \_\_position, \_Tp &&\_\_x)
- void **insert** (**iterator** \_\_p, **initializer\_list**< value\_type > \_\_l)
- void **insert** (**iterator** \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
  void **insert** (**iterator** \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **merge** (**list** &&\_\_x)
- void **merge** (**list** &\_\_x)
- template<class \_Compare >  
  void **merge** (**list** &&\_\_x, \_Compare \_\_comp)

- template<typename \_Compare >  
void **merge** (list &\_\_x, \_Compare \_\_comp)
- list & **operator=** (const list &\_\_x)
- list & **operator=** (list &&\_\_x)
- list & **operator=** (initializer\_list< value\_type > \_\_l)
- void **pop\_back** ()
- void **pop\_front** ()
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- void **remove** (const \_Tp &\_\_value)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- void **sort** ()
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **splice** (iterator \_\_position, list &&\_\_x)
- void **splice** (iterator \_\_position, list &\_\_x)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_position, list &\_\_x, iterator \_\_first, iterator \_\_last)
- void **swap** (list &\_\_x)
- void **unique** ()
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_binary\_pred)

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- unsigned int **\_M\_version**

#### Protected Member Functions

- void **\_M\_detach\_all** ()
- void **\_M\_detach\_singular** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () throw ()
- void **\_M\_revalidate\_singular** ()
- void **\_M\_swap** (\_Safe\_sequence\_base &\_\_x)

#### 4.445.1 Detailed Description

template<typename \_Tp, typename \_Allocator = std::allocator<\_Tp>>class std::\_\_debug::list< \_Tp, \_Allocator >

Class std::list with safety/checking/debug instrumentation.

Definition at line 42 of file debug/list.

## 4.445.2 Member Function Documentation

4.445.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.445.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.445.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.445.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.445.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.445.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.445.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.445.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.445.2.9 void \_\_gnu\_debug::Safe\_sequence<list<\_Tp, \_Allocator>>::M\_invalidate\_if ( \_Predicate \_\_pred )  
[inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.445.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.445.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.445.2.12 void \_\_gnu\_debug::Safe\_sequence< list< \_Tp, \_Allocator > >::M\_transfer\_from\_if ( \_Safe\_sequence< list< \_Tp, \_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### 4.445.3 Member Data Documentation

4.445.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.445.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.445.3.3 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

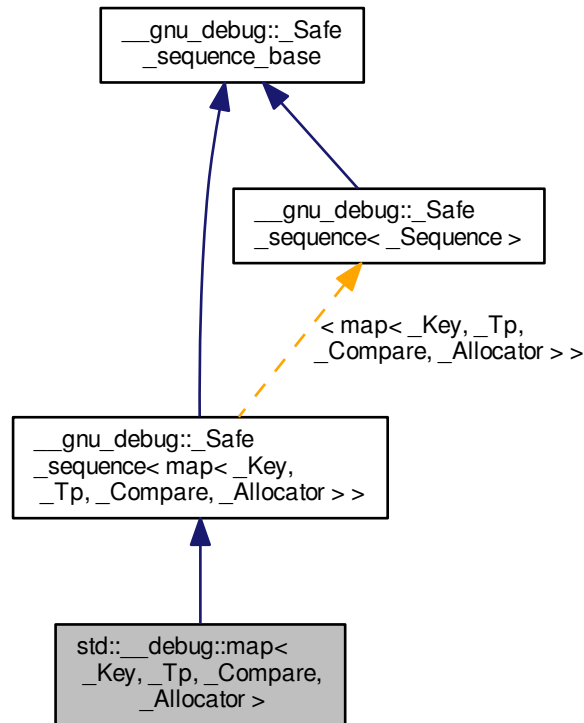
Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/list](#)

## 4.446 std::\_\_debug::map&lt; \_Key, \_Tp, \_Compare, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, map >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, map >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**

- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef std::reverse\_iterator< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef std::pair< const \_Key, \_Tp > **value\_type**

#### Public Member Functions

- **map** (const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator >  
  **map** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **map** (const map &\_\_x)
- **map** (const \_Base &\_\_x)
- **map** (map &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< \_Compare >)
- **map** (initializer\_list< value\_type > \_\_l, const \_Compare &\_\_c=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- void **\_M\_attach** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- **\_Base & \_M\_base** () noexcept
- const **\_Base & \_M\_base** () const noexcept
- void **\_M\_detach** (\_Safe\_iterator\_base \*\_\_it)
- void **\_M\_detach\_single** (\_Safe\_iterator\_base \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- **const\_reverse\_iterator crbegin** () const noexcept
- **const\_reverse\_iterator crend** () const noexcept
- template<typename... \_Args>  
  std::pair< iterator, bool > **emplace** (\_Args &&...\_\_args)
- template<typename... \_Args>  
  **iterator emplace\_hint** (const\_iterator \_\_pos, \_Args &&...\_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- **iterator erase** (const\_iterator \_\_position)
- **iterator erase** (iterator \_\_position)
- **size\_type erase** (const key\_type &\_\_x)
- **iterator erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- **iterator find** (const key\_type &\_\_x)
- **const\_iterator find** (const key\_type &\_\_x) const
- std::pair< iterator, bool > **insert** (const value\_type &\_\_x)

- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value::type>  
std::pair< iterator, bool > insert ( _Pair &&__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value::type>  
iterator insert (const_iterator __position, _Pair &&__x)`
- `template<typename _InputIterator >  
void insert ( _InputIterator __first, _InputIterator __last)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `map & operator= (const map &__x)`
- `map & operator= (map &&__x)`
- `map & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void swap (map &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

#### 4.446.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>>class std::__debug::map< _Key, _Tp, _Compare, _Allocator >`

Class std::map with safety/checking/debug instrumentation.

Definition at line 43 of file debug/map.h.

#### 4.446.2 Member Function Documentation

4.446.2.1 `void __gnu_debug::Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant )`  
[inherited]

Attach an iterator to this sequence.

4.446.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.446.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.446.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.446.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.446.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.446.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_from\_if().

4.446.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.446.2.9 void \_\_gnu\_debug::Safe\_sequence<map<\_Key, \_Tp, \_Compare, \_Allocator > >::M\_invalidate\_if ( \_Predicate  
\_\_pred ) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.446.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.446.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.446.2.12 void \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if ( \_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### 4.446.3 Member Data Documentation

4.446.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

4.446.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

4.446.3.3 unsigned int \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version [mutable],[inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

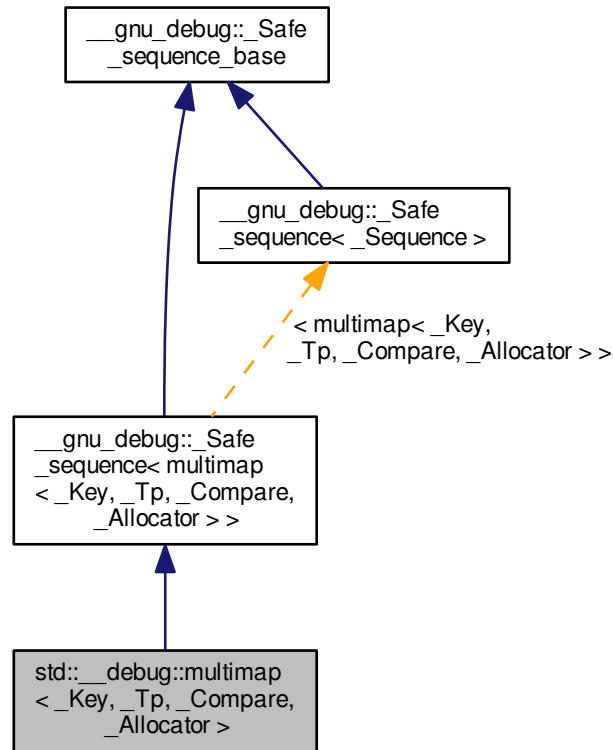
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/map.h](#)

## 4.447 std::\_\_debug::multimap&lt; \_Key, \_Tp, \_Compare, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, multimap >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, multimap >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**

- typedef \_Tp **mapped\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)  
< [iterator](#) > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key,  
\_Tp > **value\_type**

#### Public Member Functions

- **multimap** (const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator >  
**multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator  
&\_\_a=\_Allocator())
- **multimap** (const [multimap](#) &\_\_x)
- **multimap** (const [\\_Base](#) &\_\_x)
- **multimap** ([multimap](#) &&\_\_x) noexcept([is\\_nothrow\\_copy\\_constructible](#)< \_Compare >
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Compare &\_\_c=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \*\_\_it)
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \*\_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- [iterator](#) **begin** () noexcept
- const [iterator](#) **begin** () const noexcept
- const [iterator](#) **cbegin** () const noexcept
- const [iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- const [reverse\\_iterator](#) **crbegin** () const noexcept
- const [reverse\\_iterator](#) **crend** () const noexcept
- template<typename... \_Args>  
[iterator](#) **emplace** (\_Args &&...\_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_pos, \_Args &&...\_\_args)
- [iterator](#) **end** () noexcept
- const [iterator](#) **end** () const noexcept
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< const [iterator](#),  
const [iterator](#) > **equal\_range** (const key\_type &\_\_x) const
- [iterator](#) **erase** (const [iterator](#) \_\_position)
- [iterator](#) **erase** ([iterator](#) \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- [iterator](#) **erase** (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [iterator](#) **find** (const key\_type &\_\_x)
- const [iterator](#) **find** (const key\_type &\_\_x) const

- **iterator insert** (const [value\\_type](#) &\_\_x)
- **template**<typename [\\_Pair](#) , typename = typename std::enable\_if<std::is\_constructible<[value\\_type](#), [\\_Pair](#)&&>::value>::type>  
**iterator insert** ([\\_Pair](#) &&\_\_x)
- **void insert** ([std::initializer\\_list](#)< [value\\_type](#) > \_\_list)
- **iterator insert** (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- **template**<typename [\\_Pair](#) , typename = typename std::enable\_if<std::is\_constructible<[value\\_type](#), [\\_Pair](#)&&>::value>::type>  
**iterator insert** (const [iterator](#) \_\_position, [\\_Pair](#) &&\_\_x)
- **template**<typename [\\_InputIterator](#) >  
**void insert** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- **iterator lower\_bound** (const [key\\_type](#) &\_\_x)
- **const\_iterator lower\_bound** (const [key\\_type](#) &\_\_x) const
- **multimap & operator=** (const [multimap](#) &\_\_x)
- **multimap & operator=** ([multimap](#) &&\_\_x)
- **multimap & operator=** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- **reverse\_iterator rbegin** () noexcept
- **const\_reverse\_iterator rbegin** () const noexcept
- **reverse\_iterator rend** () noexcept
- **const\_reverse\_iterator rend** () const noexcept
- **void swap** ([multimap](#) &\_\_x)
- **iterator upper\_bound** (const [key\\_type](#) &\_\_x)
- **const\_iterator upper\_bound** (const [key\\_type](#) &\_\_x) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- **void** [\\_M\\_detach\\_all](#) ()
- **void** [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- **void** [\\_M\\_revalidate\\_singular](#) ()
- **void** [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

#### 4.447.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std-
::pair<const _Key, _Tp> >>> class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class std::multimap with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multimap.h.

#### 4.447.2 Member Function Documentation

4.447.2.1 **void** [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M.attach](#) ( [\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.447.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.447.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.447.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.447.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.447.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.447.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

4.447.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.447.2.9 void \_\_gnu\_debug::Safe\_sequence< multimap< \_Key, \_Tp, \_Compare, \_Allocator > >::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.447.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.447.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.447.2.12 void \_\_gnu\_debug::Safe\_sequence< multimap< \_Key, \_Tp, \_Compare, \_Allocator > >::M\_transfer\_from\_if  
( \_Safe\_sequence< multimap< \_Key, \_Tp, \_Compare, \_Allocator > > & \_\_from, \_Predicate \_\_pred )  
[inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### 4.447.3 Member Data Documentation

4.447.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.447.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.447.3.3 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

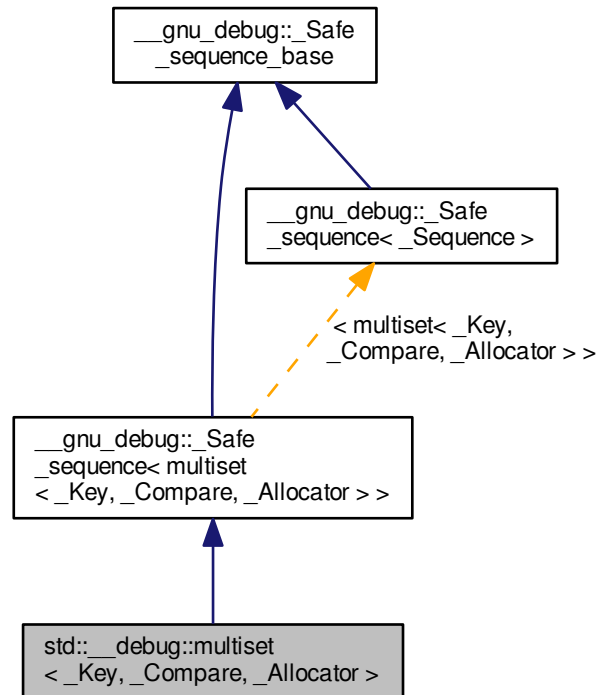
Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

4.448 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, multiset >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, multiset >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**

- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

#### Public Member Functions

- **multiset** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **multiset** (const `multiset` &\_\_x)
- **multiset** (const `_Base` &\_\_x)
- **multiset** (`multiset` &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< `_Compare` >)
- **multiset** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- void **\_M\_attach** (`_Safe_iterator_base` \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (`_Safe_iterator_base` \*\_\_it, bool \_\_constant) throw ()
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_detach** (`_Safe_iterator_base` \*\_\_it)
- void **\_M\_detach\_single** (`_Safe_iterator_base` \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- `iterator` **begin** () noexcept
- const `iterator` **begin** () const noexcept
- const `iterator` **cbegin** () const noexcept
- const `iterator` **cend** () const noexcept
- void **clear** () noexcept
- const `reverse_iterator` **crbegin** () const noexcept
- const `reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`>  
`iterator` **emplace** (`_Args` &&...\_\_args)
- template<typename... `_Args`>  
`iterator` **emplace\_hint** (const `iterator` \_\_pos, `_Args` &&...\_\_args)
- `iterator` **end** () noexcept
- const `iterator` **end** () const noexcept
- `std::pair`< `iterator`, `iterator` > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const `iterator`,  
const `iterator` > **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (const `iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator` **find** (const `key_type` &\_\_x)
- const `iterator` **find** (const `key_type` &\_\_x) const
- `iterator` **insert** (const `value_type` &\_\_x)
- `iterator` **insert** (`value_type` &&\_\_x)
- `iterator` **insert** (const `iterator` \_\_position, const `value_type` &\_\_x)

- [iterator insert](#) ([const\\_iterator](#) \_\_position, value\_type &&\_\_x)
- [template<typename \\_InputIterator > void insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [void insert](#) ([initializer\\_list](#)< value\_type > \_\_l)
- [iterator lower\\_bound](#) (const key\_type &\_\_x)
- [const\\_iterator lower\\_bound](#) (const key\_type &\_\_x) const
- [multiset & operator=](#) (const [multiset](#) &\_\_x)
- [multiset & operator=](#) ([multiset](#) &&\_\_x)
- [multiset & operator=](#) ([initializer\\_list](#)< value\_type > \_\_l)
- [reverse\\_iterator rbegin](#) () noexcept
- [const\\_reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- [void swap](#) ([multiset](#) &\_\_x)
- [iterator upper\\_bound](#) (const key\_type &\_\_x)
- [const\\_iterator upper\\_bound](#) (const key\_type &\_\_x) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- [void \\_M\\_detach\\_all](#) ()
- [void \\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex & \\_M\\_get\\_mutex](#) () throw ()
- [void \\_M\\_revalidate\\_singular](#) ()
- [void \\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

#### 4.448.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>class std::__debug::multiset<_Key, _Compare, _Allocator >`

Class std::multiset with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multiset.h.

#### 4.448.2 Member Function Documentation

4.448.2.1 `void __gnu_debug::Safe_sequence_base::M.attach ( _Safe_iterator_base * __it, bool __constant )`  
[inherited]

Attach an iterator to this sequence.

4.448.2.2 `void __gnu_debug::Safe_sequence_base::M.attach_single ( _Safe_iterator_base * __it, bool __constant ) throw ()`  
[inherited]

Likewise but not thread safe.

4.448.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.448.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.448.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.448.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.448.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_from\_if().

4.448.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.448.2.9 void \_\_gnu\_debug::Safe\_sequence< multiset<\_Key, \_Compare, \_Allocator > >::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.448.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.448.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.448.2.12 void \_\_gnu\_debug::Safe\_sequence< multiset<\_Key, \_Compare, \_Allocator > >::M\_transfer\_from\_if ( \_Safe\_sequence< multiset<\_Key, \_Compare, \_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators x that reference from sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

#### 4.448.3 Member Data Documentation

##### 4.448.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

##### 4.448.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

##### 4.448.3.3 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe\_base.h.

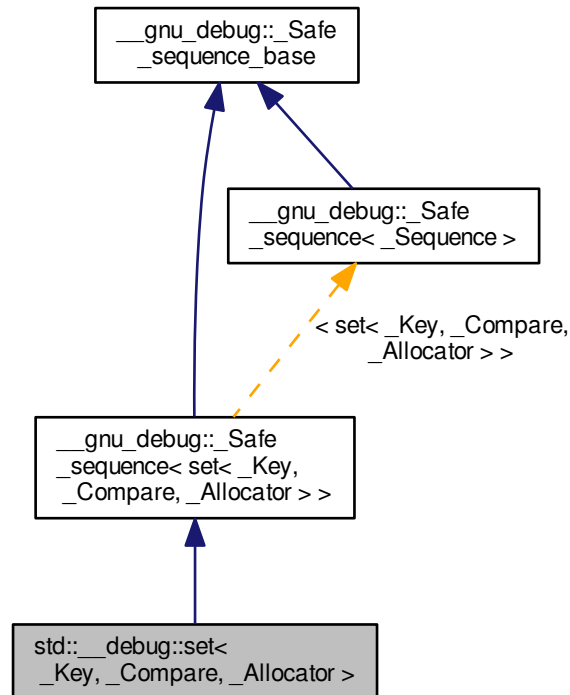
Referenced by \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

4.449 `std::__debug::set<_Key, _Compare, _Allocator>` Class Template Reference

Inheritance diagram for `std::__debug::set<_Key, _Compare, _Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, set>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, set>` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**

- typedef [std::reverse\\_iterator](#) < [iterator](#) > **reverse\_iterator**
- typedef [\\_Base::size\\_type](#) **size\_type**
- typedef [\\_Compare](#) **value\_compare**
- typedef [\\_Key](#) **value\_type**

#### Public Member Functions

- **set** (const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- template<typename [\\_InputIterator](#) >  
  **set** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **set** (const [set](#) &\_\_x)
- **set** (const [\\_Base](#) &\_\_x)
- **set** ([set](#) &&\_\_x) noexcept([is\\_nothrow\\_copy\\_constructible](#)< [\\_Compare](#) >)
- **set** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- void [\\_M\\_attach](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_detach](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it)
- void [\\_M\\_detach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \*\_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) ([\\_Safe\\_sequence](#) &\_\_from, [\\_Predicate](#) \_\_pred)
- [iterator](#) **begin** () noexcept
- const [iterator](#) **begin** () const noexcept
- const [iterator](#) **cbegin** () const noexcept
- const [iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- const [reverse\\_iterator](#) **crbegin** () const noexcept
- const [reverse\\_iterator](#) **crend** () const noexcept
- template<typename... [\\_Args](#)>  
  [std::pair](#)< [iterator](#), bool > **emplace** ([\\_Args](#) &&...\_\_args)
- template<typename... [\\_Args](#)>  
  [iterator](#) **emplace\_hint** (const [iterator](#) \_\_pos, [\\_Args](#) &&...\_\_args)
- [iterator](#) **end** () noexcept
- const [iterator](#) **end** () const noexcept
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const [key\\_type](#) &\_\_x)
- [std::pair](#)< const [iterator](#),  
  const [iterator](#) > **equal\_range** (const [key\\_type](#) &\_\_x) const
- [iterator](#) **erase** (const [iterator](#) \_\_position)
- [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
- [iterator](#) **erase** (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [iterator](#) **find** (const [key\\_type](#) &\_\_x)
- const [iterator](#) **find** (const [key\\_type](#) &\_\_x) const
- [std::pair](#)< [iterator](#), bool > **insert** (const [value\\_type](#) &\_\_x)
- [std::pair](#)< [iterator](#), bool > **insert** ([value\\_type](#) &&\_\_x)
- [iterator](#) **insert** (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- [iterator](#) **insert** (const [iterator](#) \_\_position, [value\\_type](#) &&\_\_x)

- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- set & **operator=** (const set &\_\_x)
- set & **operator=** (set &&\_\_x)
- set & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** (set &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- unsigned int **\_M\_version**

#### Protected Member Functions

- void **\_M\_detach\_all** ()
- void **\_M\_detach\_singular** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () throw ()
- void **\_M\_revalidate\_singular** ()
- void **\_M\_swap** (\_Safe\_sequence\_base &\_\_x)

#### 4.449.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>>class std::\_\_debug::set<\_Key, \_Compare, \_Allocator>

Class std::set with safety/checking/debug instrumentation.

Definition at line 43 of file debug/set.h.

#### 4.449.2 Member Function Documentation

4.449.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M.attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.449.2.2 void \_\_gnu\_debug::Safe\_sequence\_base::M.attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.449.2.3 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.449.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

4.449.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.449.2.6 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.449.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_from\_if().

4.449.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.449.2.9 void \_\_gnu\_debug::Safe\_sequence<set<\_Key, \_Compare, \_Allocator >>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.449.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.449.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.449.2.12 void \_\_gnu\_debug::Safe\_sequence<set<\_Key, \_Compare, \_Allocator >>::M\_transfer\_from\_if ( \_Safe\_sequence<set<\_Key, \_Compare, \_Allocator >> & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators x that reference from sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

## 4.449.3 Member Data Documentation

4.449.3.1 `_Safe_iterator_base*` `__gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.449.3.2 `_Safe_iterator_base*` `__gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

4.449.3.3 `unsigned int` `__gnu_debug::Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

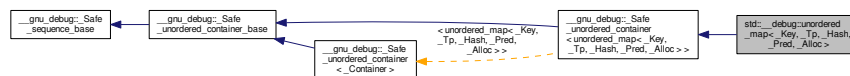
Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/set.h](#)

4.450 `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>` Class Template Reference

Inheritance diagram for `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`:



## Public Types

- `typedef _Base::allocator_type` **allocator\_type**
- `typedef`  
`__gnu_debug::Safe_iterator`  
`<_Base_const_iterator,`  
`unordered_map>` **const\_iterator**
- `typedef`  
`__gnu_debug::Safe_local_iterator`  
`<_Base_const_local_iterator,`  
`unordered_map>` **const\_local\_iterator**
- `typedef _Base::hasher` **hasher**
- `typedef`  
`__gnu_debug::Safe_iterator`  
`<_Base_iterator,`  
`unordered_map>` **iterator**

- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef  
`__gnu_debug::Safe_local_iterator`  
`<_Base_local_iterator,`  
`unordered_map >` **local\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

#### Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const [unordered\\_map](#) &\_\_x)=default
- **unordered\_map** (const [\\_Base](#) &\_\_x)
- **unordered\_map** ([unordered\\_map](#) &&\_\_x)=default
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_local](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_local\\_single](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \* \_\_it)
- void [\\_M\\_detach\\_local](#) (\_Safe\_iterator\_base \* \_\_it)
- void [\\_M\\_detach\\_local\\_single](#) (\_Safe\_iterator\_base \* \_\_it) throw ()
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \* \_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_invalidate\\_local\\_if](#) (\_Predicate \_\_pred)
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **local\_iterator begin** (size\_type \_\_b)
- **const\_local\_iterator begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- **const\_iterator cbegin** () const noexcept
- **const\_local\_iterator cbegin** (size\_type \_\_b) const
- **const\_iterator cend** () const noexcept
- **const\_local\_iterator cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
`std::pair< iterator, bool >` **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_hint, \_Args &&... \_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **local\_iterator end** (size\_type \_\_b)

- `const_local_iterator end` (size\_type \_\_b) const
- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_key)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_key) const
- size\_type `erase` (const key\_type &\_\_key)
- `iterator erase` (const\_iterator \_\_it)
- `iterator erase` (iterator \_\_it)
- `iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- `iterator find` (const key\_type &\_\_key)
- `const_iterator find` (const key\_type &\_\_key) const
- `std::pair< iterator, bool > insert` (const value\_type &\_\_obj)
- `iterator insert` (const\_iterator \_\_hint, const value\_type &\_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value::type>  
`std::pair< iterator, bool > insert` (\_Pair && \_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value::type>  
`iterator insert` (const\_iterator \_\_hint, \_Pair && \_\_obj)
- void `insert` (std::initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_f)
- `unordered_map & operator=` (const `unordered_map` &\_\_x)
- `unordered_map & operator=` (`unordered_map` && \_\_x)
- `unordered_map & operator=` (initializer\_list< value\_type > \_\_l)
- void `swap` (`unordered_map` &\_\_x)

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- unsigned int `_M_version`

#### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (\_Safe\_unordered\_container\_base &\_\_x)
- void `_M_swap` (\_Safe\_sequence\_base &\_\_x)

#### 4.450.1 Detailed Description

template<typename \_Key, typename \_Tp, typename \_Hash = std::hash<\_Key>, typename \_Pred = std::equal\_to<\_Key>, typename \_Alloc = std::allocator<\_Key>> class std::\_\_debug::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc>

Class std::unordered\_map with safety/checking/debug instrumentation.

Definition at line 50 of file debug/unordered\_map.

## 4.450.2 Member Function Documentation

4.450.2.1 void \_\_gnu\_debug::Safe\_sequence\_base::M.attach ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this sequence.

4.450.2.2 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.attach\_local ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this container.

4.450.2.3 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.attach\_local\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw () [inherited]

Likewise but not thread safe.

4.450.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M.attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.450.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M.detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.450.2.6 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.detach\_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

4.450.2.7 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.detach\_local ( \_Safe\_iterator\_base \* \_\_it )  
[inherited]

Detach an iterator from this container

4.450.2.8 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.detach\_local\_single ( \_Safe\_iterator\_base \* \_\_it ) throw ()  
[inherited]

Likewise but not thread safe.

4.450.2.9 void \_\_gnu\_debug::Safe\_sequence\_base::M.detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.450.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M.detach\_singular ( ) [protected], [inherited]

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.450.2.11 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M.get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_\_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_from\_if().

4.450.2.12 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::M\_version.

4.450.2.13 void \_\_gnu\_debug::Safe\_unordered\_container< unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > >::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.450.2.14 void \_\_gnu\_debug::Safe\_unordered\_container< unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > >::M\_invalidate\_local\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.450.2.15 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.450.2.16 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_swap ( \_Safe\_unordered\_container\_base & \_\_x ) [protected],[inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.450.2.17 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.450.3 Member Data Documentation

4.450.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

4.450.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_const\_local\_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe\_unordered\_base.h.

4.450.3.3 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe\_base.h.

## 4.451 std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference 1343

Referenced by `__gnu_debug::__Safe_sequence< _Sequence >::__M_transfer_from_if()`.

### 4.450.3.4 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::\_\_M\_local\_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

### 4.450.3.5 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::\_\_M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

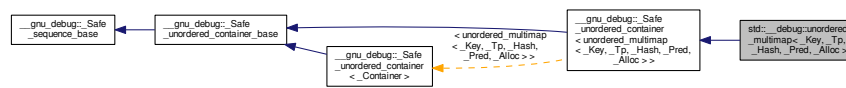
Referenced by `__gnu_debug::Safe_sequence_base::__M_invalidate_all()`, and `__gnu_debug::Safe_sequence< _Sequence >::__M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 4.451 std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, unordered_multimap >` **const\_iterator**
- typedef `__gnu_debug::Safe_local_iterator< _Base_const_local_iterator, unordered_multimap >` **const\_local\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, unordered_multimap >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `__gnu_debug::Safe_local_iterator< _Base_local_iterator, unordered_multimap >` **local\_iterator**
- typedef `_Base::size_type` **size\_type**

- `typedef _Base::value_type value_type`

#### Public Member Functions

- **`unordered_multimap`** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator>`  
**`unordered_multimap`** (`_InputIterator __first`, `_InputIterator __last`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **`unordered_multimap`** (`const unordered_multimap &__x`)=default
- **`unordered_multimap`** (`const _Base &__x`)
- **`unordered_multimap`** (`unordered_multimap &&__x`)=default
- **`unordered_multimap`** (`initializer_list<value_type> __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `_Base & _M_base ()` noexcept
- `const _Base & _M_base ()` const noexcept
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_local (_Safe_iterator_base * __it)`
- `void _M_detach_local_single (_Safe_iterator_base * __it) throw ()`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all ()` const
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_invalidate_local_if (_Predicate __pred)`
- `iterator begin ()` noexcept
- `const_iterator begin ()` const noexcept
- `local_iterator begin (size_type __b)`
- `const_local_iterator begin (size_type __b)` const
- `size_type bucket_size (size_type __b)` const
- `const_iterator cbegin ()` const noexcept
- `const_local_iterator cbegin (size_type __b)` const
- `const_iterator cend ()` const noexcept
- `const_local_iterator cend (size_type __b)` const
- `void clear ()` noexcept
- `template<typename... _Args>`  
`iterator emplace (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __hint, _Args &&... __args)`
- `iterator end ()` noexcept
- `const_iterator end ()` const noexcept
- `local_iterator end (size_type __b)`
- `const_local_iterator end (size_type __b)` const
- `std::pair<iterator, iterator> equal_range (const key_type & __key)`
- `std::pair<const_iterator, const_iterator> equal_range (const key_type & __key)` const
- `size_type erase (const key_type & __key)`
- `iterator erase (const_iterator __it)`
- `iterator erase (iterator __it)`

- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const value_type &__obj`)
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`_Pair &&__obj`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`const_iterator __hint`, `_Pair &&__obj`)
- `void insert` (`std::initializer_list<value_type> __l`)
- `template<typename _InputIterator>`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `float max_load_factor` () `const noexcept`
- `void max_load_factor` (`float __f`)
- `unordered_multimap & operator=` (`const unordered_multimap &__x`)
- `unordered_multimap & operator=` (`unordered_multimap &&__x`)
- `unordered_multimap & operator=` (`initializer_list<value_type> __l`)
- `void swap` (`unordered_multimap &__x`)

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () `throw` ()
- `void _M_revalidate_singular` ()
- `void _M_swap` (`_Safe_unordered_container_base &__x`)
- `void _M_swap` (`_Safe_sequence_base &__x`)

#### 4.451.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Definition at line 478 of file `debug/unordered_map`.

#### 4.451.2 Member Function Documentation

4.451.2.1 `void __gnu_debug::Safe_sequence_base::M_attach` (`_Safe_iterator_base * __it`, `bool __constant`)  
[inherited]

Attach an iterator to this sequence.

4.451.2.2 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.attach\_local ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant )  
[inherited]

Attach an iterator to this container.

4.451.2.3 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.attach\_local\_single ( \_Safe\_iterator\_base \* \_\_it, bool  
\_\_constant ) throw () [inherited]

Likewise but not thread safe.

4.451.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M.attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

4.451.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M.detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.451.2.6 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

4.451.2.7 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.detach\_local ( \_Safe\_iterator\_base \* \_\_it )  
[inherited]

Detach an iterator from this container

4.451.2.8 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M.detach\_local\_single ( \_Safe\_iterator\_base \* \_\_it ) throw ()  
[inherited]

Likewise but not thread safe.

4.451.2.9 void \_\_gnu\_debug::Safe\_sequence\_base::M.detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.451.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M.detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.451.2.11 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M.get\_mutex ( ) throw () [protected],  
[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.451.2.12 void \_\_gnu\_debug::Safe\_sequence\_base::M.invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.451.2.13 void \_\_gnu\_debug::Safe\_unordered\_container<unordered\_multimap<\_Key,\_Tp,\_Hash,\_Pred,\_Alloc>::\_\_M\_invalidate\_if( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.451.2.14 void \_\_gnu\_debug::Safe\_unordered\_container<unordered\_multimap<\_Key,\_Tp,\_Hash,\_Pred,\_Alloc>::\_\_M\_invalidate\_local\_if( \_Predicate \_\_pred ) [inherited]

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.451.2.15 void \_\_gnu\_debug::Safe\_sequence\_base::\_\_M\_revalidate\_singular( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.451.2.16 void \_\_gnu\_debug::Safe\_unordered\_container\_base::\_\_M\_swap( \_Safe\_unordered\_container\_base & \_\_x ) [protected],[inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.451.2.17 void \_\_gnu\_debug::Safe\_sequence\_base::\_\_M\_swap( \_Safe\_sequence\_base & \_\_x ) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.451.3 Member Data Documentation

4.451.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_\_M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::__M_transfer_from_if()`.

4.451.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::\_\_M\_const\_local\_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe\_unordered\_base.h.

4.451.3.3 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_\_M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::__M_transfer_from_if()`.

4.451.3.4 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::\_\_M\_local\_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe\_unordered\_base.h.

4.451.3.5 `unsigned int __gnu_debug::Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

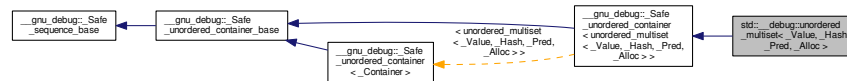
Referenced by `__gnu_debug::Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 4.452 `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Inheritance diagram for `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`:



### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef`  
`__gnu_debug::Safe_iterator`  
`< _Base_const_iterator,`  
`unordered_multiset > const_iterator`
- `typedef`  
`__gnu_debug::Safe_local_iterator`  
`< _Base_const_local_iterator,`  
`unordered_multiset > const_local_iterator`
- `typedef _Base::hasher hasher`
- `typedef`  
`__gnu_debug::Safe_iterator`  
`< _Base_iterator,`  
`unordered_multiset > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef`  
`__gnu_debug::Safe_local_iterator`  
`< _Base_local_iterator,`  
`unordered_multiset > local_iterator`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- **unordered\_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)

- `template<typename _InputIterator >`  
`unordered_multiset` (`_InputIterator __first`, `_InputIterator __last`, `size_type __n=0`, `const hasher &__hf=hasher()`,  
`const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_multiset` (`const unordered_multiset &__x`)=default
- `unordered_multiset` (`const _Base &__x`)
- `unordered_multiset` (`unordered_multiset &&__x`)=default
- `unordered_multiset` (`initializer_list< value_type > __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const`  
`key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `void _M_attach` (`_Safe_iterator_base *__it`, `bool __constant`)
- `void _M_attach_local` (`_Safe_iterator_base *__it`, `bool __constant`)
- `void _M_attach_local_single` (`_Safe_iterator_base *__it`, `bool __constant`) `throw ()`
- `void _M_attach_single` (`_Safe_iterator_base *__it`, `bool __constant`) `throw ()`
- `_Base & _M_base` () `noexcept`
- `const _Base & _M_base` () `const noexcept`
- `void _M_detach` (`_Safe_iterator_base *__it`)
- `void _M_detach_local` (`_Safe_iterator_base *__it`)
- `void _M_detach_local_single` (`_Safe_iterator_base *__it`) `throw ()`
- `void _M_detach_single` (`_Safe_iterator_base *__it`) `throw ()`
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_invalidate_local_if` (`_Predicate __pred`)
- `iterator begin` () `noexcept`
- `const_iterator begin` () `const noexcept`
- `local_iterator begin` (`size_type __b`)
- `const_local_iterator begin` (`size_type __b`) `const`
- `size_type bucket_size` (`size_type __b`) `const`
- `const_iterator cbegin` () `const noexcept`
- `const_local_iterator cbegin` (`size_type __b`) `const`
- `const_iterator cend` () `const noexcept`
- `const_local_iterator cend` (`size_type __b`) `const`
- `void clear` () `noexcept`
- `template<typename... _Args>`  
`iterator emplace` (`_Args &&... __args`)
- `template<typename... _Args>`  
`iterator emplace_hint` (`const_iterator __hint`, `_Args &&... __args`)
- `iterator end` () `noexcept`
- `const_iterator end` () `const noexcept`
- `local_iterator end` (`size_type __b`)
- `const_local_iterator end` (`size_type __b`) `const`
- `std::pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `std::pair< const_iterator,`  
`const_iterator > equal_range` (`const key_type &__key`) `const`
- `size_type erase` (`const key_type &__key`)
- `iterator erase` (`const_iterator __it`)
- `iterator erase` (`iterator __it`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const value_type &__obj`)
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `iterator insert` (`value_type &&__obj`)

- [iterator insert](#) ([const\\_iterator](#) \_\_hint, [value\\_type](#) &&\_\_obj)
- [void insert](#) ([std::initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [template](#)<typename [\\_InputIterator](#) >  
[void insert](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- [float max\\_load\\_factor](#) () [const noexcept](#)
- [void max\\_load\\_factor](#) ([float](#) \_\_f)
- [unordered\\_multiset](#) & [operator=](#) ([const unordered\\_multiset](#) &\_\_x)
- [unordered\\_multiset](#) & [operator=](#) ([unordered\\_multiset](#) &&\_\_x)
- [unordered\\_multiset](#) & [operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [void swap](#) ([unordered\\_multiset](#) &\_\_x)

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_local\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_local\\_iterators](#)
- [unsigned int](#) [\\_M\\_version](#)

#### Protected Member Functions

- [void](#) [\\_M\\_detach\\_all](#) ()
- [void](#) [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () [throw](#) ()
- [void](#) [\\_M\\_revalidate\\_singular](#) ()
- [void](#) [\\_M\\_swap](#) ([\\_Safe\\_unordered\\_container\\_base](#) &\_\_x)
- [void](#) [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

#### 4.452.1 Detailed Description

`template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>> class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 474 of file `debug/unordered_set`.

#### 4.452.2 Member Function Documentation

4.452.2.1 `void __gnu_debug::Safe_sequence_base::M.attach ( _Safe_iterator_base * __it, bool __constant )`  
[[inherited](#)]

Attach an iterator to this sequence.

4.452.2.2 `void __gnu_debug::Safe_unordered_container_base::M.attach_local ( _Safe_iterator_base * __it, bool __constant )`  
[[inherited](#)]

Attach an iterator to this container.

4.452.2.3 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_attach\_local\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw () [inherited]

Likewise but not thread safe.

4.452.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw () [inherited]

Likewise but not thread safe.

4.452.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.452.2.6 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

4.452.2.7 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_detach\_local ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this container

4.452.2.8 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_detach\_local\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.452.2.9 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.452.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.452.2.11 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

4.452.2.12 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.452.2.13 void \_\_gnu\_debug::Safe\_unordered\_container< unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >  
>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.452.2.14 void \_\_gnu\_debug::Safe\_unordered\_container< unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >  
>::M\_invalidate\_local\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.452.2.15 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.452.2.16 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_swap ( \_Safe\_unordered\_container\_base & \_\_x )  
[protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.452.2.17 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.452.3 Member Data Documentation

4.452.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.452.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_const\_local\_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe\_unordered\_base.h.

4.452.3.3 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.452.3.4 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_local\_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe\_unordered\_base.h.

4.452.3.5 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe\_base.h.

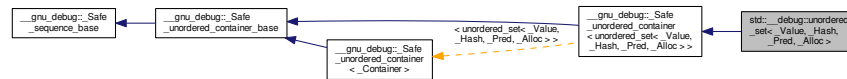
Referenced by \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 4.453 std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inheritance diagram for std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >:



### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#) < \_Base\_const\_iterator, [unordered\\_set](#) > **const\_iterator**
- typedef [\\_\\_gnu\\_debug::Safe\\_local\\_iterator](#) < \_Base\_const\_local\_iterator, [unordered\\_set](#) > **const\_local\_iterator**
- typedef \_Base::hasher **hasher**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#) < \_Base\_iterator, [unordered\\_set](#) > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_local\\_iterator](#) < \_Base\_local\_iterator, [unordered\\_set](#) > **local\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

### Public Member Functions

- **unordered\_set** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())

- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const [unordered\\_set](#) &\_\_x)=default
- **unordered\_set** (const [\\_Base](#) &\_\_x)
- **unordered\_set** ([unordered\\_set](#) && \_\_x)=default
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_local](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_local\\_single](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \* \_\_it)
- void [\\_M\\_detach\\_local](#) (\_Safe\_iterator\_base \* \_\_it)
- void [\\_M\\_detach\\_local\\_single](#) (\_Safe\_iterator\_base \* \_\_it) throw ()
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \* \_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_invalidate\\_local\\_if](#) (\_Predicate \_\_pred)
- [iterator](#) **begin** () noexcept
- const [iterator](#) **begin** () const noexcept
- [local\\_iterator](#) **begin** (size\_type \_\_b)
- const [local\\_iterator](#) **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const [iterator](#) **cbegin** () const noexcept
- const [local\\_iterator](#) **cbegin** (size\_type \_\_b) const
- const [iterator](#) **cend** () const noexcept
- const [local\\_iterator](#) **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_hint, \_Args &&... \_\_args)
- [iterator](#) **end** () noexcept
- const [iterator](#) **end** () const noexcept
- [local\\_iterator](#) **end** (size\_type \_\_b)
- const [local\\_iterator](#) **end** (size\_type \_\_b) const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_key)
- [std::pair](#)< const [iterator](#),  
const [iterator](#) > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- [iterator](#) **erase** (const [iterator](#) \_\_it)
- [iterator](#) **erase** ([iterator](#) \_\_it)
- [iterator](#) **erase** (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [iterator](#) **find** (const key\_type &\_\_key)
- const [iterator](#) **find** (const key\_type &\_\_key) const
- [std::pair](#)< [iterator](#), bool > **insert** (const value\_type &\_\_obj)
- [iterator](#) **insert** (const [iterator](#) \_\_hint, const value\_type &\_\_obj)
- [std::pair](#)< [iterator](#), bool > **insert** (value\_type && \_\_obj)

- `iterator insert` (`const_iterator __hint`, `value_type &&__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `float max_load_factor` () `const noexcept`
- `void max_load_factor` (`float __f`)
- `unordered_set & operator=` (`const unordered_set &__x`)
- `unordered_set & operator=` (`unordered_set &&__x`)
- `unordered_set & operator=` (`initializer_list< value_type > __l`)
- `void swap` (`unordered_set &__x`)

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () `throw` ()
- `void _M_revalidate_singular` ()
- `void _M_swap` (`_Safe_unordered_container_base &__x`)
- `void _M_swap` (`_Safe_sequence_base &__x`)

#### 4.453.1 Detailed Description

`template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>> class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`

Class `std::unordered_set` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_set`.

#### 4.453.2 Member Function Documentation

4.453.2.1 `void __gnu_debug::Safe_sequence_base::M.attach ( _Safe_iterator_base * __it, bool __constant )`  
[inherited]

Attach an iterator to this sequence.

4.453.2.2 `void __gnu_debug::Safe_unordered_container_base::M.attach_local ( _Safe_iterator_base * __it, bool __constant )`  
[inherited]

Attach an iterator to this container.

4.453.2.3 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_attach\_local\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw () [inherited]

Likewise but not thread safe.

4.453.2.4 void \_\_gnu\_debug::Safe\_sequence\_base::M\_attach\_single ( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw () [inherited]

Likewise but not thread safe.

4.453.2.5 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

4.453.2.6 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_detach\_all ( ) [protected],[inherited]

Detach all iterators, leaving them singular.

4.453.2.7 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_detach\_local ( \_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this container

4.453.2.8 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_detach\_local\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.453.2.9 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single ( \_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

4.453.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( ) [protected],[inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

4.453.2.11 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_base::M\_get\_mutex ( ) throw () [protected],[inherited]

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.453.2.12 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

4.453.2.13 void \_\_gnu\_debug::Safe\_unordered\_container< unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >  
>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.453.2.14 void \_\_gnu\_debug::Safe\_unordered\_container< unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >  
>::M\_invalidate\_local\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.453.2.15 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.453.2.16 void \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_swap ( \_Safe\_unordered\_container\_base & \_\_x )  
[protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.453.2.17 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],  
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.453.3 Member Data Documentation

4.453.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.453.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_const\_local\_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file `safe_unordered_base.h`.

4.453.3.3 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_from_if()`.

4.453.3.4 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_unordered\_container\_base::M\_local\_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

4.453.3.5 unsigned int \_\_gnu\_debug::\_\_Safe\_sequence\_base::\_M\_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe\_base.h.

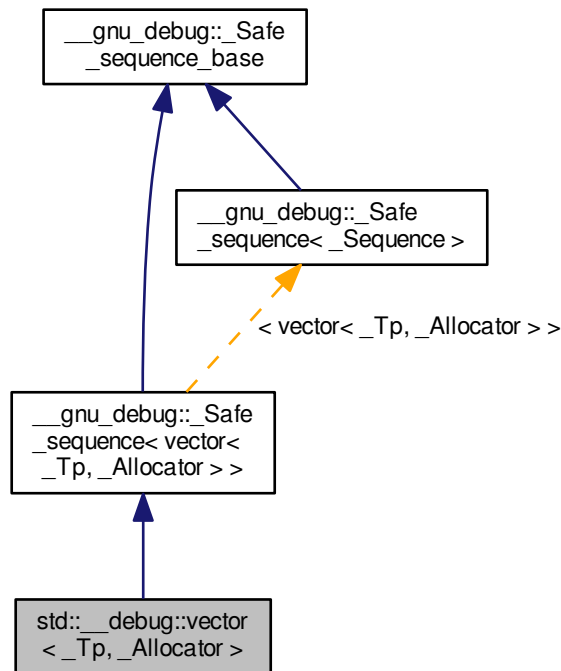
Referenced by \_\_gnu\_debug::\_\_Safe\_sequence\_base::\_M\_invalidate\_all(), and \_\_gnu\_debug::\_\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 4.454 std::\_\_debug::vector< \_Tp, \_Allocator > Class Template Reference

Inheritance diagram for std::\_\_debug::vector< \_Tp, \_Allocator >:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, vector >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**

- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)  
< [const\\_iterator](#) > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef  
[\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)  
< [\\_Base\\_iterator](#), [vector](#) > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)  
< [iterator](#) > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **vector** (const \_Allocator &\_\_a= \_Allocator())
- **vector** (size\_type \_\_n, const \_Allocator &\_\_a= \_Allocator())
- **vector** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a= \_Allocator())
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>  
**vector** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a= \_Allocator())
- **vector** (const [vector](#) &\_\_x)
- [vector](#) (const [\\_Base](#) &\_\_x)
- **vector** ([vector](#) &&\_\_x) noexcept
- **vector** (const [vector](#) &\_\_x, const allocator\_type &\_\_a)
- **vector** ([vector](#) &&\_\_x, const allocator\_type &\_\_a)
- **vector** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \* \_\_it)
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \* \_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence & \_\_from, \_Predicate \_\_pred)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_u)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- reference **back** ()
- const\_reference **back** () const
- [iterator](#) **begin** () noexcept
- [const\\_iterator](#) **begin** () const noexcept
- size\_type **capacity** () const noexcept
- [const\\_iterator](#) **cbegin** () const noexcept
- [const\\_iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept

- template<typename... \_Args>  
    **iterator emplace** (**iterator** \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
    void **emplace\_back** (\_Args &&...\_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **iterator erase** (**iterator** \_\_position)
- **iterator erase** (**iterator** \_\_first, **iterator** \_\_last)
- reference **front** ()
- const\_reference **front** () const
- **iterator insert** (**iterator** \_\_position, const \_Tp &\_\_x)
- template<typename \_Up = \_Tp>  
    \_\_gnu\_cxx::\_\_enable\_if  
    <!std::\_\_are\_same<\_Up, bool>  
    ::\_\_value, **iterator**>::\_\_type **insert** (**iterator** \_\_position, \_Tp &&\_\_x)
- void **insert** (**iterator** \_\_position, **initializer\_list**<value\_type> \_\_l)
- void **insert** (**iterator** \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>  
    void **insert** (**iterator** \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- **vector** & **operator=** (const **vector** &\_\_x)
- **vector** & **operator=** (**vector** &&\_\_x) noexcept(**\_Alloc\_traits**
- **vector** & **operator=** (**initializer\_list**<value\_type> \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (const \_Tp &\_\_x)
- template<typename \_Up = \_Tp>  
    \_\_gnu\_cxx::\_\_enable\_if  
    <!std::\_\_are\_same<\_Up, bool>  
    ::\_\_value, void>::\_\_type **push\_back** (\_Tp &&\_\_x)
- **reverse\_iterator rbegin** () noexcept
- **const\_reverse\_iterator rbegin** () const noexcept
- **reverse\_iterator rend** () noexcept
- **const\_reverse\_iterator rend** () const noexcept
- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- void **shrink\_to\_fit** ()
- void **swap** (**vector** &\_\_x) noexcept(**\_Alloc\_traits**

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- unsigned int **\_M\_version**

#### Protected Member Functions

- void **\_M\_detach\_all** ()
- void **\_M\_detach\_singular** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () throw ()
- void **\_M\_revalidate\_singular** ()
- void **\_M\_swap** (\_Safe\_sequence\_base &\_\_x)

## 4.454.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__debug::vector<_Tp, _Allocator >
```

Class std::vector with safety/checking/debug instrumentation.

Definition at line 44 of file debug/vector.

## 4.454.2 Constructor &amp; Destructor Documentation

```
4.454.2.1 template<typename _Tp, typename _Allocator = std::allocator<_Tp>> std::__debug::vector<_Tp, _Allocator>::vector (const _Base & __x) [inline]
```

Construction from a release-mode vector.

Definition at line 115 of file debug/vector.

## 4.454.3 Member Function Documentation

```
4.454.3.1 void __gnu_debug::Safe_sequence_base::M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]
```

Attach an iterator to this sequence.

```
4.454.3.2 void __gnu_debug::Safe_sequence_base::M_attach_single (_Safe_iterator_base * __it, bool __constant) throw () [inherited]
```

Likewise but not thread safe.

```
4.454.3.3 void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it) [inherited]
```

Detach an iterator from this sequence

```
4.454.3.4 void __gnu_debug::Safe_sequence_base::M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

```
4.454.3.5 void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw () [inherited]
```

Likewise but not thread safe.

```
4.454.3.6 void __gnu_debug::Safe_sequence_base::M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

```
4.454.3.7 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw () [protected], [inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_from\_if().

4.454.3.8 void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( ) const [inline],[inherited]

Invalidates all iterators.

Definition at line 233 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::M\_version.

4.454.3.9 void \_\_gnu\_debug::Safe\_sequence<vector<\_Tp, \_Allocator>>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which \_\_pred(*x*) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

4.454.3.10 void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.454.3.11 void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.454.3.12 void \_\_gnu\_debug::Safe\_sequence<vector<\_Tp, \_Allocator>>::M\_transfer\_from\_if ( \_Safe\_sequence<vector<\_Tp, \_Allocator>> & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which \_\_pred(*x*) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

#### 4.454.4 Member Data Documentation

4.454.4.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.454.4.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

4.454.4.3 unsigned int \_\_gnu\_debug::Safe\_sequence\_base::M\_version [mutable],[inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe\_base.h.

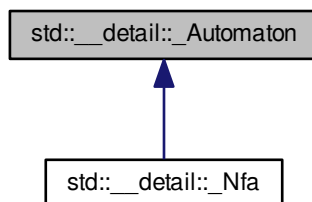
Referenced by \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all(), and \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

The documentation for this class was generated from the following file:

- [debug/vector](#)

#### 4.455 `std::__detail::_Automaton` Class Reference

Inheritance diagram for `std::__detail::_Automaton`:



##### Public Types

- typedef unsigned int **\_SizeT**

##### Public Member Functions

- virtual `_SizeT` **\_M\_sub\_count** () const =0

##### 4.455.1 Detailed Description

Base class for, um, automata. Could be an NFA or a DFA. Your choice.

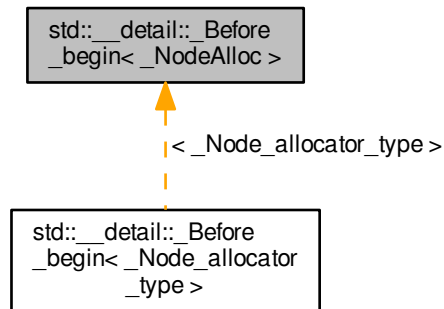
Definition at line 43 of file `regex_nfa.h`.

The documentation for this class was generated from the following file:

- [regex\\_nfa.h](#)

## 4.456 std::\_\_detail::\_Before\_begin&lt; \_NodeAlloc &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Before\_begin< \_NodeAlloc >:



## Public Member Functions

- `_Before_begin` (const `_Before_begin` &)=default
- `_Before_begin` (`_Before_begin` &&)=default
- `template<typename _Alloc > _Before_begin` (`_Alloc` &&\_\_a)

## Public Attributes

- `_Hash_node_base` `_M_node`

## 4.456.1 Detailed Description

```
template<typename _NodeAlloc>struct std::__detail::_Before_begin< _NodeAlloc >
```

This type is to combine a `_Hash_node_base` instance with an allocator instance through inheritance to benefit from EBO when possible.

Definition at line 1652 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- `hashtable_policy.h`

## 4.457 std::\_\_detail::\_CharMatcher&lt; \_InlterT, \_TraitsT &gt; Struct Template Reference

## Public Types

- `typedef _TraitsT::char_type` `char_type`

## Public Member Functions

- **\_CharMatcher** (char\_type \_\_c, const \_TraitsT &\_\_t=\_TraitsT())
- bool **operator()** (const [\\_PatternCursor](#) &\_\_pc) const

## Public Attributes

- char\_type **\_M\_c**
- const \_TraitsT & **\_M\_traits**

## 4.457.1 Detailed Description

```
template<typename _InIterT, typename _TraitsT> struct std::__detail::_CharMatcher< _InIterT, _TraitsT >
```

Matches a single character.

Definition at line 129 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

4.458 `std::__detail::_Compiler<_InIter, _TraitsT>` Class Template Reference

## Public Types

- typedef std::iterator\_traits<\_InIter>::value\_type **\_CharT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef \_InIter **\_IterT**
- typedef [std::basic\\_string](#)<\_CharT> **\_StringT**

## Public Member Functions

- **\_Compiler** (const \_InIter &\_\_b, const \_InIter &\_\_e, \_TraitsT &\_\_traits, \_FlagT \_\_flags)
- const [\\_Nfa](#) & **\_M\_nfa** () const

## 4.458.1 Detailed Description

```
template<typename _InIter, typename _TraitsT> class std::__detail::_Compiler< _InIter, _TraitsT >
```

Builds an NFA from an input iterator interval.

Definition at line 634 of file `regex_compiler.h`.

The documentation for this class was generated from the following file:

- [regex\\_compiler.h](#)

## 4.459 std::\_\_detail::\_\_Default\_ranged\_hash Struct Reference

## 4.459.1 Detailed Description

Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that  $h(k, N) = h2(h1(k), N)$ , but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.

Definition at line 353 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.460 std::\_\_detail::\_\_EndTagger&lt;\_FwdIterT, \_TraitsT&gt; Struct Template Reference

## Public Member Functions

- **\_EndTagger** (int \_\_i)
- void **operator()** (const [\\_PatternCursor](#) &\_\_pc, [\\_Results](#) &\_\_r)

## Public Attributes

- int **\_M\_index**
- [\\_FwdIterT](#) **\_M\_pos**

## 4.460.1 Detailed Description

```
template<typename _FwdIterT, typename _TraitsT> struct std::__detail::__EndTagger<_FwdIterT, _TraitsT>
```

End state tag.

Definition at line 104 of file regex\_nfa.h.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

## 4.461 std::\_\_detail::\_\_Equal\_helper&lt;\_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, false&gt; Struct Template Reference

## Static Public Member Functions

- static bool **\_S\_equals** (const [\\_Equal](#) &\_\_eq, const [\\_ExtractKey](#) &\_\_extract, const [\\_Key](#) &\_\_k, [\\_HashCodeType](#), [\\_Hash\\_node](#)<\_Value, false> \*\_\_n)

## 4.461.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType> struct std::__detail::__Equal_helper<_Key, _Value, _ExtractKey, _Equal, _HashCodeType, false>
```

Specialization.

Definition at line 1172 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.462 **std::\_\_detail::\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, true > Struct Template Reference**

##### Static Public Member Functions

- static bool **\_S\_equals** (const \_Equal &\_\_eq, const \_ExtractKey &\_\_extract, const \_Key &\_\_k, \_HashCodeType \_\_c, [\\_Hash\\_node](#)< \_Value, true > \*\_\_n)

##### 4.462.1 Detailed Description

template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_Equal, typename \_HashCodeType>struct std::\_\_detail::\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, true >

Specialization.

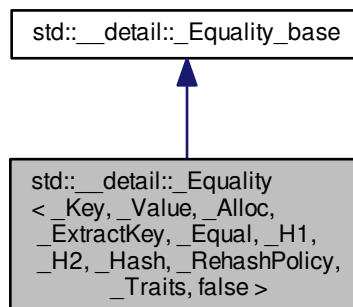
Definition at line 1161 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.463 **std::\_\_detail::\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false > Struct Template Reference**

Inheritance diagram for std::\_\_detail::\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >:



#### Public Types

- using `__hashtable` = `__Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

#### Public Member Functions

- `bool _M_equal` (const `__hashtable` &) const

#### Static Protected Member Functions

- `template<typename _Uiterator >`  
`static bool _S_is_permutation` (\_Uiterator, \_Uiterator, \_Uiterator)

##### 4.463.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> struct std::detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`

Specialization.

Definition at line 1604 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.464 `std::detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

#### Public Types

- using `__hashtable` = `__Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

#### Public Member Functions

- `bool _M_equal` (const `__hashtable` &) const

##### 4.464.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> struct std::detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`

Specialization.

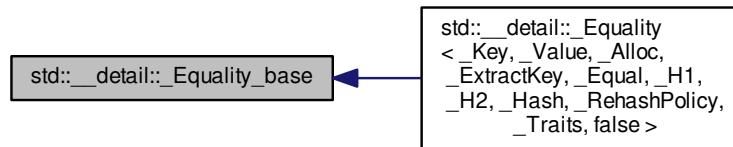
Definition at line 1566 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.465 std::\_\_detail::\_Equality\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Equality\_base:



## Static Protected Member Functions

- `template<typename _Uiterator >`  
`static bool _S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`

## 4.465.1 Detailed Description

struct `_Equality_base`.

Common types and functions for class `_Equality`.

Definition at line 1492 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.466 std::\_\_detail::\_Grep\_matcher Class Reference

## Public Member Functions

- `_Grep_matcher (\_PatternCursor &__p, \_Results &__r, const \_AutomatonPtr &__automaton, regex\_constants::match\_flag\_type __flags)`

## 4.466.1 Detailed Description

Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the `grep` utility, modified to use Laurikari tags.

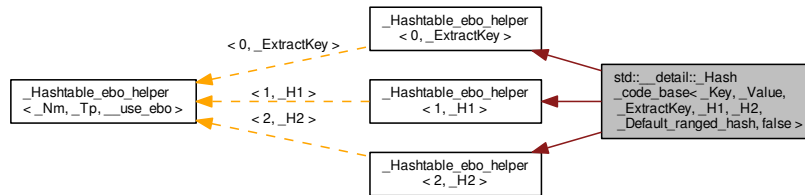
Definition at line 110 of file `regex_grep_matcher.h`.

The documentation for this class was generated from the following files:

- [regex\\_grep\\_matcher.h](#)
- [regex\\_grep\\_matcher.tcc](#)

#### 4.467 `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >` **Struct Template Reference**

Inheritance diagram for `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >`:



#### Public Types

- typedef `_H1 hasher`

#### Public Member Functions

- hasher **hash\_function** () const

#### Protected Types

- typedef `std::size_t __hash_code`
- typedef `_Hash_node< _Value, false > __node_type`

#### Protected Member Functions

- **\_Hash\_code\_base** (const `_ExtractKey` &\_\_ex, const `_H1` &\_\_h1, const `_H2` &\_\_h2, const `_Default_ranged_hash` &)
- `std::size_t _M_bucket_index` (const `_Key` &, `__hash_code` \_\_c, `std::size_t` \_\_n) const
- `std::size_t _M_bucket_index` (const `__node_type` \*\_\_p, `std::size_t` \_\_n) const
- void **\_M\_copy\_code** (`__node_type` \*, const `__node_type` \*) const
- const `_ExtractKey` & **\_M\_extract** () const
- `_ExtractKey` & **\_M\_extract** ()
- const `_H1` & **\_M\_h1** () const
- `_H1` & **\_M\_h1** ()
- const `_H2` & **\_M\_h2** () const
- `_H2` & **\_M\_h2** ()
- `__hash_code` **\_M\_hash\_code** (const `_Key` &\_\_k) const
- void **\_M\_store\_code** (`__node_type` \*, `__hash_code`) const
- void **\_M\_swap** (`_Hash_code_base` &\_\_x)

#### 4.467.1 Detailed Description

`template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2> struct std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >`

Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.

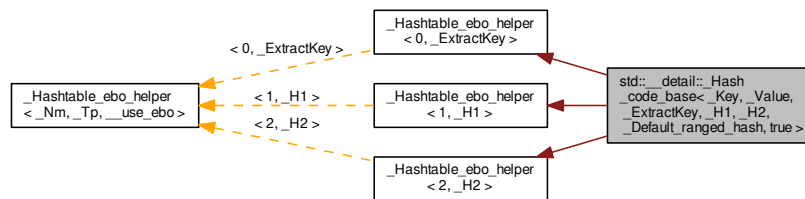
Definition at line 987 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.468 `std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >` **Struct Template Reference**

Inheritance diagram for `std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`:



#### Public Types

- typedef `_H1` **hasher**

#### Public Member Functions

- hasher **hash\_function** () const

#### Protected Types

- typedef `std::size_t` **\_\_hash\_code**
- typedef `_Hash_node<_Value, true >` **\_\_node\_type**

#### Protected Member Functions

- **\_Hash\_code\_base** (const `_ExtractKey` &\_\_ex, const `_H1` &\_\_h1, const `_H2` &\_\_h2, const `_Default_ranged_hash` &\_\_)
- `std::size_t` **\_M\_bucket\_index** (const `_Key` &\_\_, `__hash_code` \_\_c, `std::size_t` \_\_n) const
- `std::size_t` **\_M\_bucket\_index** (const `__node_type` \*\_\_p, `std::size_t` \_\_n) const

- `void _M_copy_code ( __node_type * __to, const __node_type * __from) const`
- `const _ExtractKey & _M_extract () const`
- `_ExtractKey & _M_extract ()`
- `const _H1 & _M_h1 () const`
- `_H1 & _M_h1 ()`
- `const _H2 & _M_h2 () const`
- `_H2 & _M_h2 ()`
- `__hash_code _M_hash_code (const _Key & __k) const`
- `void _M_store_code ( __node_type * __n, __hash_code __c) const`
- `void _M_swap ( _Hash_code_base & __x)`

#### Friends

- `struct _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`

#### 4.468.1 Detailed Description

`template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2> struct std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`

Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.

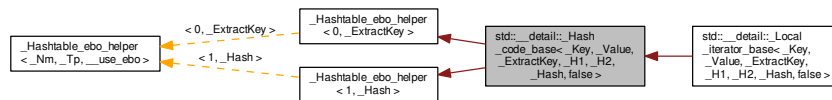
Definition at line 1070 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.469 `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`:



#### Protected Types

- `typedef void * __hash_code`
- `typedef _Hash_node<_Value, false > __node_type`

## Protected Member Functions

- **\_Hash\_code\_base** (const \_ExtractKey &\_\_ex, const \_H1 &, const \_H2 &, const \_Hash &\_\_h)
- std::size\_t **\_M\_bucket\_index** (const \_Key &\_\_k, \_\_hash\_code, std::size\_t \_\_n) const
- std::size\_t **\_M\_bucket\_index** (const \_\_node\_type \*\_\_p, std::size\_t \_\_n) const
- void **\_M\_copy\_code** (\_\_node\_type \*, const \_\_node\_type \*) const
- const \_ExtractKey & **\_M\_extract** () const
- \_ExtractKey & **\_M\_extract** ()
- \_\_hash\_code **\_M\_hash\_code** (const \_Key &\_\_key) const
- const \_Hash & **\_M\_ranged\_hash** () const
- \_Hash & **\_M\_ranged\_hash** ()
- void **\_M\_store\_code** (\_\_node\_type \*, \_\_hash\_code) const
- void **\_M\_swap** (\_Hash\_code\_base &\_\_x)

## 4.469.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

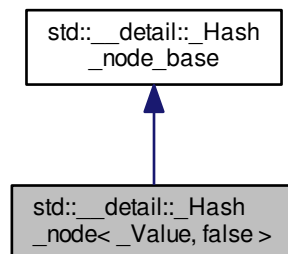
Definition at line 913 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.470 std::\_\_detail::\_Hash\_node&lt; \_Value, false &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node< \_Value, false >:



## Public Member Functions

- template<typename... \_Args>  
  **\_Hash\_node** (\_Args &&... \_\_args)
- [\\_Hash\\_node](#) \* **\_M\_next** () const

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* `_M_nxt`
- `_Value` `_M_v`

## 4.470.1 Detailed Description

```
template<typename _Value>struct std::__detail::_Hash_node< _Value, false >
```

Specialization for nodes without caches, struct `_Hash_node`.

Base class is `__detail::_Hash_node_base`.

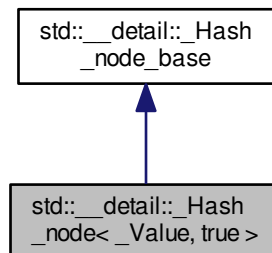
Definition at line 189 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.471 `std::__detail::_Hash_node< _Value, true >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node< _Value, true >`:



## Public Member Functions

- `template<typename... _Args>`  
`_Hash_node` (`_Args` &&... `__args`)
- [\\_Hash\\_node](#) \* `_M_next` () const

## Public Attributes

- `std::size_t` `_M_hash_code`
- [\\_Hash\\_node\\_base](#) \* `_M_nxt`
- `_Value` `_M_v`

## 4.471.1 Detailed Description

```
template<typename _Value>struct std::__detail::_Hash_node< _Value, true >
```

Specialization for nodes with caches, struct \_Hash\_node.

Base class is \_\_detail::\_Hash\_node\_base.

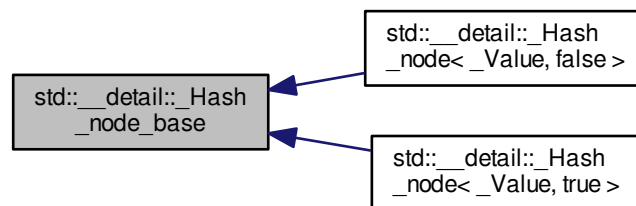
Definition at line 170 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.472 std::\_\_detail::\_Hash\_node\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node\_base:



## Public Member Functions

- [\\_Hash\\_node\\_base](#) ([\\_Hash\\_node\\_base](#) \* \_\_next)

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* [\\_M\\_nxt](#)

## 4.472.1 Detailed Description

```
struct _Hash_node_base
```

Nodes, used to wrap elements stored in the hash table. A policy template parameter of class template \_Hashtable controls whether nodes also store a hash code. In some cases (e.g. strings) this may be a performance win.

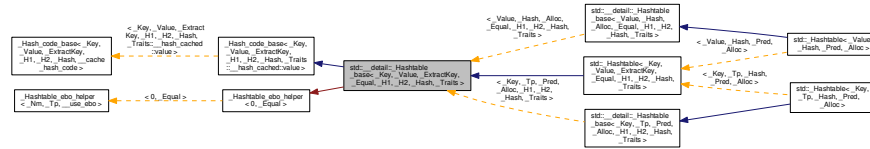
Definition at line 149 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.473 `std::detail::Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits>` Struct Template Reference

Inheritance diagram for `std::detail::Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits>` >:



### Public Types

- using **\_\_constant\_iterators** = typename `__traits_type::__constant_iterators`
- using **\_\_hash\_cached** = typename `__traits_type::__hash_cached`
- using **\_\_hash\_code** = typename `__hash_code_base::__hash_code`
- using **\_\_hash\_code\_base** = `__Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __hash_cached::value>`
- using **\_\_iconv\_type** = typename `std::conditional<__unique_keys::value, _Select1st, _Identity>::type`
- using **\_\_ireturn\_type** = typename `std::conditional<__unique_keys::value, std::pair<iterator, bool>, iterator>::type`
- using **\_\_node\_type** = typename `__hash_code_base::__node_type`
- using **\_\_traits\_type** = `_Traits`
- using **\_\_unique\_keys** = typename `__traits_type::__unique_keys`
- using **const\_iterator** = `__detail::Node_const_iterator<value_type, __constant_iterators::value, __hash_cached::value>`
- using **const\_local\_iterator** = `__detail::Local_const_iterator<key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __constant_iterators::value, __hash_cached::value>`
- typedef `std::ptrdiff_t` **difference\_type**
- using **iterator** = `__detail::Node_iterator<value_type, __constant_iterators::value, __hash_cached::value>`
- typedef `_Equal` **key\_equal**
- typedef `_Key` **key\_type**
- using **local\_iterator** = `__detail::Local_iterator<key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __constant_iterators::value, __hash_cached::value>`
- typedef `std::size_t` **size\_type**
- typedef `_Value` **value\_type**

### Protected Types

- using **\_\_bucket\_type** = `__node_base *`
- using **\_\_node\_base** = `__detail::Hash_node_base`

### Protected Member Functions

- **\_\_Hashtable\_base** (const `_ExtractKey` &\_\_ex, const `_H1` &\_\_h1, const `_H2` &\_\_h2, const `_Hash` &\_\_hash, const `_Equal` &\_\_eq)
- const `_Equal` & **\_M\_eq** () const

- `_Equal & _M_eq ()`
- `bool _M_equals (const _Key &__k, __hash_code __c, __node_type *__n) const`
- `void _M_swap (_Hashtable_base &__x)`

#### 4.473.1 Detailed Description

`template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _Traits> struct std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`

Primary class template `_Hashtable_base`.

Helper class adding management of `_Equal` functor to `_Hash_code_base` type.

Base class templates are:

- `__detail::_Hash_code_base`
- `__detail::_Hashtable_ebo_helper`

Definition at line 1402 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.474 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false >` Struct Template Reference

### Public Member Functions

- `_Hashtable_ebo_helper (const _Tp &__tp)`

### Static Public Member Functions

- `static const _Tp & _S_cget (const _Hashtable_ebo_helper &__eboh)`
- `static _Tp & _S_get (_Hashtable_ebo_helper &__eboh)`

#### 4.474.1 Detailed Description

`template<int _Nm, typename _Tp> struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >`

Specialization not using EBO.

Definition at line 854 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.475 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, true >` Struct Template Reference

Inherits `_Tp`.

## Public Member Functions

- `_Hashtable_ebo_helper` (const `_Tp` &\_\_tp)

## Static Public Member Functions

- static const `_Tp` & `_S_cget` (const `_Hashtable_ebo_helper` &\_\_eboh)
- static `_Tp` & `_S_get` (`_Hashtable_ebo_helper` &\_\_eboh)

### 4.475.1 Detailed Description

template<int `_Nm`, typename `_Tp`>struct std::\_\_detail::\_Hashtable\_ebo\_helper< `_Nm`, `_Tp`, true >

Specialization using EBO.

Definition at line 835 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.476 `std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >` Struct Template Reference

### Public Types

- template<bool `_Cond`>  
using `__bool_constant` = [integral\\_constant](#)< bool, `_Cond` >
- using `__constant_iterators` = [\\_\\_bool\\_constant](#)< `_Constant_iterators` >
- using `__hash_cached` = [\\_\\_bool\\_constant](#)< `_Cache_hash_code` >
- using `__unique_keys` = [\\_\\_bool\\_constant](#)< `_Unique_keys` >

### 4.476.1 Detailed Description

template<bool `_Cache_hash_code`, bool `_Constant_iterators`, bool `_Unique_keys`>struct std::\_\_detail::\_Hashtable\_traits< `_Cache_hash_code`, `_Constant_iterators`, `_Unique_keys` >

struct `_Hashtable_traits`

Important traits for hash tables.

### Template Parameters

|                                  |                                                                                                                                                                                                                                                                                                          |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Cache_hash_code</code>    | Boolean value. True if the value of the hash function is stored along with the value. This is a time-space tradeoff. Storing it may improve lookup speed by reducing the number of times we need to call the <code>_Equal</code> function.                                                               |
| <code>_Constant_iterators</code> | Boolean value. True if iterator and const_iterator are both constant iterator types. This is true for <code>unordered_set</code> and <code>unordered_multiset</code> , false for <code>unordered_map</code> and <code>unordered_multimap</code> .                                                        |
| <code>_Unique_keys</code>        | Boolean value. True if the return value of <code>_Hashtable::count(k)</code> is always at most one, false if it may be an arbitrary number. This is true for <code>unordered_set</code> and <code>unordered_map</code> , false for <code>unordered_multiset</code> and <code>unordered_multimap</code> . |

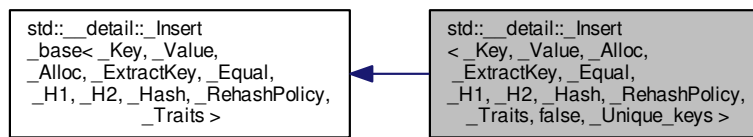
Definition at line 131 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.477 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys>` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys>`:



#### Public Types

- using `__base_type` = `_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits>`
- using `__hashtable` = `typename __base_type::__hashtable`
- using `__hashtable_base` = `_Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits>`
- using `__iconv_type` = `typename __base_type::__iconv_type`
- using `__ireturn_type` = `typename __base_type::__ireturn_type`
- template<typename `_Pair`>
  - using `__is_cons` = `std::is_constructible<value_type, _Pair &&>`
- using `__unique_keys` = `typename __base_type::__unique_keys`
- template<typename `_Pair`>
  - using `__IFcons` = `std::enable_if<__is_cons<_Pair>::value>`
- template<typename `_Pair`>
  - using `__IFconsp` = `typename __IFcons<_Pair>::type`
- using `const_iterator` = `typename __base_type::const_iterator`
- using `iterator` = `typename __base_type::iterator`
- using `size_type` = `typename __hashtable_base::size_type`
- using `value_type` = `typename __base_type::value_type`

#### Public Member Functions

- `__hashtable & __M_conjure_hashtable()`
- `__ireturn_type insert(const value_type &v)`
- `iterator insert(const_iterator, const value_type &v)`
- `void insert(initializer_list<value_type> __l)`
- template<typename `_InputIterator`>
  - `void insert(_InputIterator __first, _InputIterator __last)`

- `template<typename _Pair, typename = _IFconsp<_Pair>>`  
`__ireturn_type insert (_Pair &&__v)`
- `template<typename _Pair, typename = _IFconsp<_Pair>>`  
`iterator insert (const_iterator, _Pair &&__v)`

#### 4.477.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys> struct std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >`

Specialization.

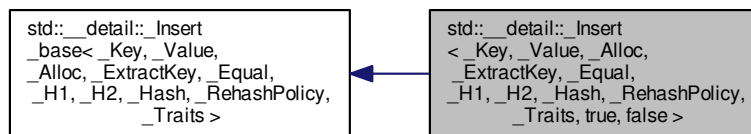
Definition at line 736 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.478 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >`:



#### Public Types

- using `__base_type` = `_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- using `__hashtable` = `typename __base_type::__hashtable`
- using `__hashtable_base` = `_Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`
- using `__iconv_type` = `typename __hashtable_base::__iconv_type`
- using `__ireturn_type` = `typename __hashtable_base::__ireturn_type`
- using `__unique_keys` = `typename __base_type::__unique_keys`
- using `const_iterator` = `typename __base_type::const_iterator`
- using `iterator` = `typename __base_type::iterator`
- using `size_type` = `typename __hashtable_base::size_type`
- using `value_type` = `typename __base_type::value_type`

#### Public Member Functions

- [\\_\\_hashtable](#) & [\\_M\\_conjure\\_hashtable](#) ()
- `__ireturn_type insert` (const value\_type &\_\_v)
- iterator `insert` (const\_iterator, const value\_type &\_\_v)
- void `insert` ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator `insert` (value\_type &&\_\_v)
- iterator `insert` (const\_iterator, value\_type &&\_\_v)

#### 4.478.1 Detailed Description

template<typename \_Key, typename \_Value, typename \_Alloc, typename \_ExtractKey, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_RehashPolicy, typename \_Traits>struct std::\_\_detail::Insert<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true, false >

Specialization.

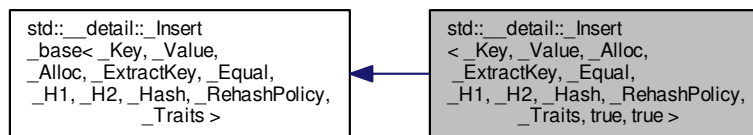
Definition at line 702 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.479 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >`:



#### Public Types

- using `__base_type` = [\\_Insert\\_base](#)<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using `__hashtable` = typename [\\_\\_base\\_type::\\_\\_hashtable](#)
- using `__hashtable_base` = [Hashtable\\_base](#)<\_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using `__iconv_type` = typename `__hashtable_base::__iconv_type`
- using `__ireturn_type` = typename `__hashtable_base::__ireturn_type`
- using `__unique_keys` = typename `__base_type::__unique_keys`

- using **const\_iterator** = typename \_\_base\_type::const\_iterator
- using **iterator** = typename \_\_base\_type::iterator
- using **size\_type** = typename \_\_hashtable\_base::size\_type
- using **value\_type** = typename \_\_base\_type::value\_type

#### Public Member Functions

- [\\_\\_hashtable](#) & **\_M\_conjure\_hashtable** ()
- `__ireturn_type` **insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator, const value\_type &\_\_v)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_v)
- iterator **insert** (const\_iterator, value\_type &&\_\_v)

#### 4.479.1 Detailed Description

template<typename \_Key, typename \_Value, typename \_Alloc, typename \_ExtractKey, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_RehashPolicy, typename \_Traits>struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true, true >

Specialization.

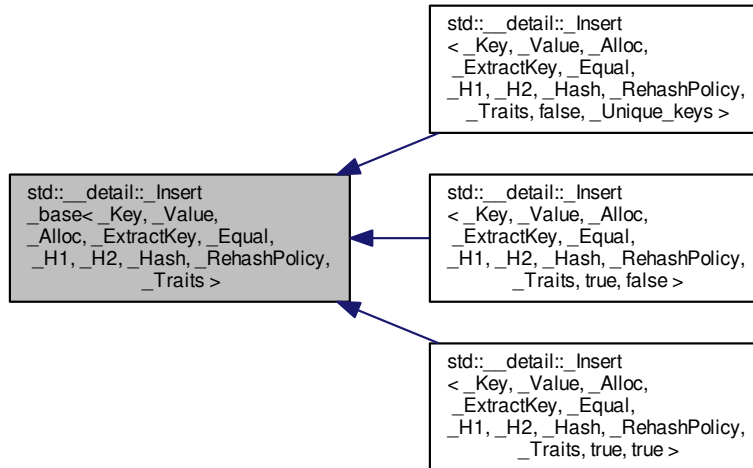
Definition at line 668 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.480 std::\_\_detail::\_\_Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_\_Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >:



#### Public Types

- using **\_\_hashtable** = [\\_Hashtable](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using **\_\_hashtable\_base** = [\\_Hashtable\\_base](#)< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using **\_\_iconv\_type** = typename \_\_hashtable\_base::\_\_iconv\_type
- using **\_\_ireturn\_type** = typename \_\_hashtable\_base::\_\_ireturn\_type
- using **\_\_unique\_keys** = typename \_\_hashtable\_base::\_\_unique\_keys
- using **const\_iterator** = typename \_\_hashtable\_base::const\_iterator
- using **iterator** = typename \_\_hashtable\_base::iterator
- using **size\_type** = typename \_\_hashtable\_base::size\_type
- using **value\_type** = typename \_\_hashtable\_base::value\_type

#### Public Member Functions

- [\\_hashtable](#) & **\_M\_conjure\_hashtable** ()
- \_\_ireturn\_type **insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator, const value\_type &\_\_v)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)

## 4.480.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> struct std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Insert_base`.

insert member functions appropriate to all `_Hashtables`.

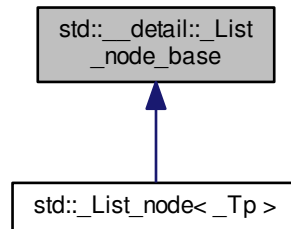
Definition at line 577 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.481 std::\_\_detail::\_List\_node\_base Struct Reference

Inheritance diagram for `std::__detail::_List_node_base`:



## Public Member Functions

- void **\_M\_hook** ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept

## Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

## Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 4.481.1 Detailed Description

Common part of a node in the list.

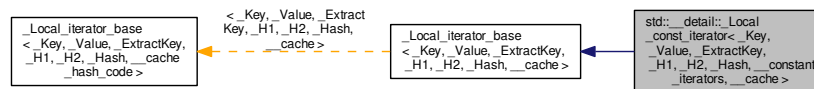
Definition at line 77 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 4.482 `std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

#### Public Member Functions

- `_Local_const_iterator` (`const __hash_code_base &__base, \_Hash\_node< _Value, __cache > *__p, std::size_t __bkt, std::size_t __bkt_count`)
- `_Local_const_iterator` (`const \_Local\_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > &__x`)
- reference **operator\*** () const
- `\_Local\_const\_iterator & operator++ ()`
- `\_Local\_const\_iterator operator++ (int)`
- pointer **operator->** () const

#### 4.482.1 Detailed Description

`template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache> struct std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`

`local_const_iterators`

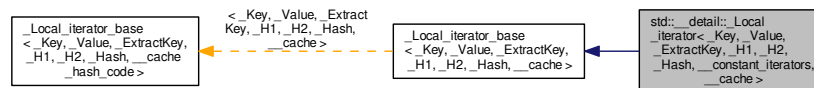
Definition at line 1334 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.483 `std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef [std::conditional](#)  
`< __constant_iterators, const  
 _Value *, _Value * >::type` **pointer**
- typedef [std::conditional](#)  
`< __constant_iterators, const  
 _Value &, _Value & >::type` **reference**
- typedef `_Value` **value\_type**

#### Public Member Functions

- **\_Local\_iterator** (const `__hash_code_base &__base`, [\\_Hash\\_node](#)< `_Value`, `__cache` > \*`__p`, `std::size_t __bkt`, `std::size_t __bkt_count`)
- reference **operator\*** () const
- [\\_Local\\_iterator](#) & **operator++** ()
- [\\_Local\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

#### 4.483.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache> struct std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >
```

local iterators

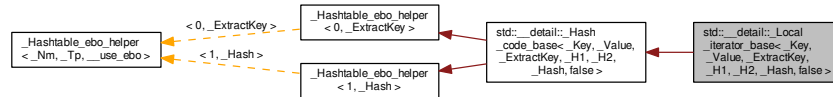
Definition at line 1279 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.484 `std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`:



## Public Member Functions

- `_Local_iterator_base` (const `__hash_code_base` &\_\_base, `_Hash_node`< `_Value`, false > \*\_\_p, `std::size_t` \_\_bkt, `std::size_t` \_\_bkt\_count)
- `void _M_incr` ()

## Public Attributes

- `std::size_t` `_M_bucket`
- `std::size_t` `_M_bucket_count`
- `_Hash_node`< `_Value`, false > \* `_M_cur`

## Protected Types

- using `__hash_code_base` = `_Hash_code_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Hash`, false >

## Private Types

- typedef void \* `__hash_code`
- typedef `_Hash_node`< `_Value`, false > `__node_type`

## Private Member Functions

- `std::size_t` `_M_bucket_index` (const `_Key` &\_\_k, `__hash_code`, `std::size_t` \_\_n) const
- `std::size_t` `_M_bucket_index` (const `__node_type` \*\_\_p, `std::size_t` \_\_n) const
- `void` `_M_copy_code` (`__node_type` \*, const `__node_type` \*) const
- const `_ExtractKey` & `_M_extract` () const
- `_ExtractKey` & `_M_extract` ()
- `__hash_code` `_M_hash_code` (const `_Key` &\_\_key) const
- const `_Hash` & `_M_ranged_hash` () const
- `_Hash` & `_M_ranged_hash` ()
- `void` `_M_store_code` (`__node_type` \*, `__hash_code`) const
- `void` `_M_swap` (`_Hash_code_base` &\_\_x)

#### 4.484.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>struct std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization.

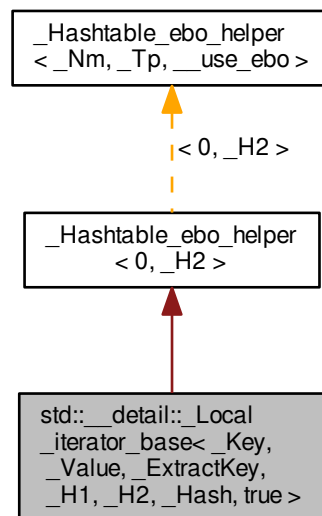
Definition at line 1223 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.485 std::\_\_detail::\_\_Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, true > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_\_Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, true >:



#### Public Member Functions

- `_Local_iterator_base` (const [\\_\\_hash\\_code\\_base](#) &\_\_base, [\\_Hash\\_node](#)< \_Value, true > \*\_\_p, std::size\_t \_\_bkt, std::size\_t \_\_bkt\_count)
- `void _M_incr ()`

#### Public Attributes

- `std::size_t _M_bucket`
- `std::size_t _M_bucket_count`



4.487 `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

Public Types

- using **mapped\_type** = typename std::tuple\_element< 1, \_Pair >::type

4.487.1 Detailed Description

template<typename \_Key, typename \_Pair, typename \_Alloc, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_RehashPolicy, typename \_Traits>struct std::\_\_detail::Map\_base<\_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >

Partial specialization, \_\_unique\_keys set to false.

Definition at line 430 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.488 `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

Public Types

- using **iterator** = typename \_\_hashtable\_base::iterator
- using **key\_type** = typename \_\_hashtable\_base::key\_type
- using **mapped\_type** = typename std::tuple\_element< 1, \_Pair >::type

Public Member Functions

- mapped\_type & **at** (const key\_type &\_\_k)
- const mapped\_type & **at** (const key\_type &\_\_k) const
- mapped\_type & **operator[]** (const key\_type &\_\_k)
- mapped\_type & **operator[]** (key\_type &&\_\_k)

4.488.1 Detailed Description

template<typename \_Key, typename \_Pair, typename \_Alloc, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_RehashPolicy, typename \_Traits>struct std::\_\_detail::Map\_base<\_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >

Partial specialization, \_\_unique\_keys set to true.

Definition at line 440 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.489 std::\_\_detail::\_\_Mod\_range\_hashing Struct Reference

### Public Types

- typedef std::size\_t **first\_argument\_type**
- typedef std::size\_t **result\_type**
- typedef std::size\_t **second\_argument\_type**

### Public Member Functions

- result\_type **operator()** (first\_argument\_type \_\_num, second\_argument\_type \_\_den) const

#### 4.489.1 Detailed Description

Default range hashing function: use division to fold a large number into the range [0, N).

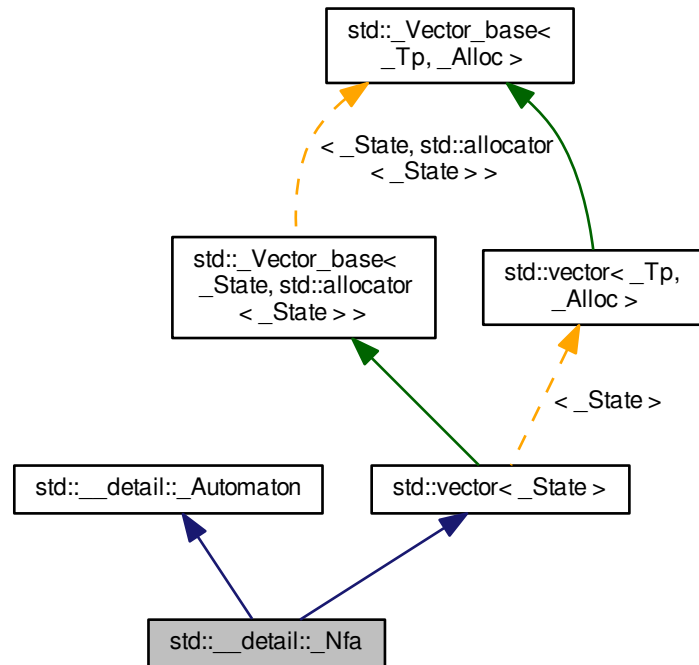
Definition at line 337 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.490 std::\_\_detail::\_Nfa Class Reference

Inheritance diagram for std::\_\_detail::\_Nfa:



## Public Types

- typedef `regex_constants::syntax_option_type` **\_FlagT**
- typedef unsigned int **\_SizeT**
- typedef `_State` **\_StateT**
- typedef `std::allocator<_State>` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**

- typedef `_Base::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator`  
    < iterator > **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_State` **value\_type**

#### Public Member Functions

- `_Nfa` (`_FlagT` \_\_f)
- `const _StateSet & _M_final_states` () const
- `_StateldT _M_insert_accept` ()
- `_StateldT _M_insert_alt` (`_StateldT` \_\_next, `_StateldT` \_\_alt)
- `_StateldT _M_insert_matcher` (`_Matcher` \_\_m)
- `_StateldT _M_insert_subexpr_begin` (const `_Tagger` &\_\_t)
- `_StateldT _M_insert_subexpr_end` (unsigned int \_\_i, const `_Tagger` &\_\_t)
- `_FlagT _M_options` () const
- `_StateldT _M_start` () const
- `_SizeT _M_sub_count` () const
- `void assign` (size\_type \_\_n, const value\_type &\_\_val)
- `void assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `void assign` (`initializer_list`< value\_type > \_\_l)
- `reference at` (size\_type \_\_n)
- `const_reference at` (size\_type \_\_n) const
- `reference back` ()
- `const_reference back` () const
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `size_type capacity` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- `_State * data` () noexcept
- `const _State * data` () const noexcept
- `iterator emplace` (iterator \_\_position, `_Args` &&... \_\_args)
- `void emplace_back` (`_Args` &&... \_\_args)
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `iterator erase` (iterator \_\_position)
- `iterator erase` (iterator \_\_first, iterator \_\_last)
- `reference front` ()
- `const_reference front` () const
- `iterator insert` (iterator \_\_position, const value\_type &\_\_x)
- `iterator insert` (iterator \_\_position, value\_type &&\_\_x)
- `void insert` (iterator \_\_position, `initializer_list`< value\_type > \_\_l)
- `void insert` (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- `void insert` (iterator \_\_position, `_InputIterator` \_\_first, `_InputIterator` \_\_last)

- size\_type [max\\_size](#) () const noexcept
- reference [operator\[\]](#) (size\_type \_\_n)
- const\_reference [operator\[\]](#) (size\_type \_\_n) const
- void [pop\\_back](#) ()
- void [push\\_back](#) (const value\_type &\_\_x)
- void [push\\_back](#) (value\_type &&\_\_x)
- [reverse\\_iterator](#) [rbegin](#) () noexcept
- [const\\_reverse\\_iterator](#) [rbegin](#) () const noexcept
- [reverse\\_iterator](#) [rend](#) () noexcept
- [const\\_reverse\\_iterator](#) [rend](#) () const noexcept
- void [reserve](#) (size\_type \_\_n)
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- void [swap](#) (vector &\_\_x) noexcept([\\_Alloc\\_traits](#)

#### Protected Member Functions

- pointer [\\_M\\_allocate](#) (size\_t \_\_n)
- pointer [\\_M\\_allocate\\_and\\_copy](#) (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- void [\\_M\\_assign\\_aux](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void [\\_M\\_assign\\_aux](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void [\\_M\\_assign\\_dispatch](#) (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void [\\_M\\_assign\\_dispatch](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type [\\_M\\_check\\_len](#) (size\_type \_\_n, const char \*\_\_s) const
- void [\\_M\\_deallocate](#) (pointer \_\_p, size\_t \_\_n)
- void [\\_M\\_default\\_append](#) (size\_type \_\_n)
- void [\\_M\\_default\\_initialize](#) (size\_type \_\_n)
- void [\\_M\\_emplace\\_back\\_aux](#) (\_Args &&... \_\_args)
- void [\\_M\\_erase\\_at\\_end](#) (pointer \_\_pos)
- void [\\_M\\_fill\\_assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- void [\\_M\\_fill\\_initialize](#) (size\_type \_\_n, const value\_type &\_\_value)
- void [\\_M\\_fill\\_insert](#) (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- \_Tp\_alloc\_type & [\\_M\\_get\\_Tp\\_allocator](#) () noexcept
- const \_Tp\_alloc\_type & [\\_M\\_get\\_Tp\\_allocator](#) () const noexcept
- void [\\_M\\_initialize\\_dispatch](#) (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- void [\\_M\\_initialize\\_dispatch](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void [\\_M\\_insert\\_aux](#) (iterator \_\_position, \_Args &&... \_\_args)
- void [\\_M\\_insert\\_dispatch](#) (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void [\\_M\\_insert\\_dispatch](#) (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void [\\_M\\_range\\_check](#) (size\_type \_\_n) const
- void [\\_M\\_range\\_initialize](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void [\\_M\\_range\\_initialize](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void [\\_M\\_range\\_insert](#) (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void [\\_M\\_range\\_insert](#) (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- bool [\\_M\\_shrink\\_to\\_fit](#) ()
- [allocator\\_type](#) [get\\_allocator](#) () const noexcept

## Protected Attributes

- `_Vector_impl_M_impl`

## 4.490.1 Detailed Description

struct `_Nfa`

A collection of all states making up an NFA.

An NFA is a 4-tuple  $M = (K, S, s, F)$ , where  $K$  is a finite set of states,  $S$  is the alphabet of the NFA,  $s$  is the initial state,  $F$  is a set of final (accepting) states.

This NFA class is templated on `S`, a type that will hold values of the underlying alphabet (without regard to semantics of that alphabet). The other elements of the tuple are generated during construction of the NFA and are available through accessor member functions.

Definition at line 269 of file `regex_nfa.h`.

## 4.490.2 Member Function Documentation

**4.490.2.1** `pointer std::vector<_State, std::allocator<_State>>::M_allocate_and_copy ( size_type __n, _ForwardIterator __first, _ForwardIterator __last ) [inline], [protected], [inherited]`

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

Definition at line 1135 of file `stl_vector.h`.

**4.490.2.2** `void std::vector<_State, std::allocator<_State>>::M_range_check ( size_type __n ) const [inline], [protected], [inherited]`

Safety check used only from `at()`.

Definition at line 791 of file `stl_vector.h`.

**4.490.2.3** `void std::vector<_State, std::allocator<_State>>::assign ( size_type __n, const value_type & __val ) [inline], [inherited]`

Assigns a given value to a vector.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 479 of file `stl_vector.h`.

**4.490.2.4** `void std::vector<_State, std::allocator<_State>>::assign ( _InputIterator __first, _InputIterator __last ) [inline], [inherited]`

Assigns a range to a vector.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a vector with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 498 of file `stl_vector.h`.

**4.490.2.5** `void std::vector<_State, std::allocator<_State>>::assign ( initializer_list<value_type> & __l )`  
`[inline], [inherited]`

Assigns an initializer list to a vector.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 524 of file `stl_vector.h`.

**4.490.2.6** `reference std::vector<_State, std::allocator<_State>>::at ( size_type __n )` `[inline], [inherited]`

Provides access to the data contained in the vector.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

## Returns

Read/write reference to data.

## Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 810 of file `stl_vector.h`.

**4.490.2.7** `const_reference std::vector<_State, std::allocator<_State>>::at ( size_type __n ) const` `[inline], [inherited]`

Provides access to the data contained in the vector.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

**Returns**

Read-only (constant) reference to data.

**Exceptions**

|                          |                                    |
|--------------------------|------------------------------------|
| <i>std::out_of_range</i> | If <i>__n</i> is an invalid index. |
|--------------------------|------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

Definition at line 828 of file *stl\_vector.h*.

#### 4.490.2.8 reference **std::vector<\_State, std::allocator<\_State>>::back ( )** [inline], [inherited]

Returns a read/write reference to the data at the last element of the vector.

Definition at line 855 of file *stl\_vector.h*.

#### 4.490.2.9 const\_reference **std::vector<\_State, std::allocator<\_State>>::back ( ) const** [inline], [inherited]

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 863 of file *stl\_vector.h*.

#### 4.490.2.10 iterator **std::vector<\_State, std::allocator<\_State>>::begin ( )** [inline], [noexcept], [inherited]

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 538 of file *stl\_vector.h*.

#### 4.490.2.11 const\_iterator **std::vector<\_State, std::allocator<\_State>>::begin ( ) const** [inline], [noexcept], [inherited]

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file *stl\_vector.h*.

#### 4.490.2.12 size\_type **std::vector<\_State, std::allocator<\_State>>::capacity ( ) const** [inline], [noexcept], [inherited]

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 725 of file *stl\_vector.h*.

#### 4.490.2.13 const\_iterator **std::vector<\_State, std::allocator<\_State>>::cbegin ( ) const** [inline], [noexcept], [inherited]

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 611 of file *stl\_vector.h*.

**4.490.2.14** `const_iterator std::vector<_State, std::allocator<_State>>::cend( ) const` `[inline]`,  
`[noexcept], [inherited]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file `stl_vector.h`.

**4.490.2.15** `void std::vector<_State, std::allocator<_State>>::clear( )` `[inline]`, `[noexcept]`,  
`[inherited]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1125 of file `stl_vector.h`.

**4.490.2.16** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::crbegin( ) const` `[inline]`,  
`[noexcept], [inherited]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 629 of file `stl_vector.h`.

**4.490.2.17** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::crend( ) const` `[inline]`,  
`[noexcept], [inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file `stl_vector.h`.

**4.490.2.18** `_State * std::vector<_State, std::allocator<_State>>::data( )` `[inline]`, `[noexcept]`,  
`[inherited]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 878 of file `stl_vector.h`.

**4.490.2.19** `iterator std::vector<_State, std::allocator<_State>>::emplace( iterator __position, _Args &&... __args )`  
`[inherited]`

Inserts an object in vector before specified iterator.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__args</code>     | Arguments.                   |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

**4.490.2.20** `bool std::vector<_State, std::allocator<_State>>::empty( ) const` `[inline], [noexcept], [inherited]`

Returns true if the vector is empty. (Thus begin() would equal end().)

Definition at line 734 of file stl\_vector.h.

**4.490.2.21** `iterator std::vector<_State, std::allocator<_State>>::end( )` `[inline], [noexcept], [inherited]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file stl\_vector.h.

**4.490.2.22** `const_iterator std::vector<_State, std::allocator<_State>>::end( ) const` `[inline], [noexcept], [inherited]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file stl\_vector.h.

**4.490.2.23** `iterator std::vector<_State, std::allocator<_State>>::erase( iterator __position )` `[inherited]`

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.490.2.24** `iterator std::vector<_State, std::allocator<_State>>::erase( iterator __first, iterator __last )` `[inherited]`

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**4.490.2.25** `reference std::vector<_State, std::allocator<_State>>::front ( )` `[inline],[inherited]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 839 of file `stl_vector.h`.

**4.490.2.26** `const_reference std::vector<_State, std::allocator<_State>>::front ( ) const` `[inline],[inherited]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 847 of file `stl_vector.h`.

**4.490.2.27** `iterator std::vector<_State, std::allocator<_State>>::insert ( iterator __position, const value_type & __x )` `[inherited]`

Inserts given value into vector before specified iterator.

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__x</code>        | Data to be inserted.         |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**4.490.2.28** `iterator std::vector<_State, std::allocator<_State>>::insert ( iterator __position, value_type && __x )` `[inline],[inherited]`

Inserts given rvalue into vector before specified iterator.

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__x</code>        | Data to be inserted.         |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 988 of file `stl_vector.h`.

4.490.2.29 `void std::vector<_State, std::allocator<_State>>::insert ( iterator __position, initializer_list< value_type > __l ) [inline], [inherited]`

Inserts an initializer\_list into the vector.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__l</code>        | An initializer_list.         |

This function will insert copies of the data in the initializer\_list *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1005 of file `stl_vector.h`.

4.490.2.30 `void std::vector<_State, std::allocator<_State>>::insert ( iterator __position, size_type __n, const value_type & __x ) [inline], [inherited]`

Inserts a number of copies of given data into the vector.

#### Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the vector.       |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1023 of file `stl_vector.h`.

4.490.2.31 `void std::vector<_State, std::allocator<_State>>::insert ( iterator __position, InputIterator __first, InputIterator __last ) [inline], [inherited]`

Inserts a range into the vector.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__first</code>    | An input iterator.           |
| <code>__last</code>     | An input iterator.           |

This function will insert copies of the data in the range [`__first`,`__last`) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1044 of file `stl_vector.h`.

4.490.2.32 `size_type std::vector<_State, std::allocator<_State>>::max_size ( ) const [inline], [noexcept], [inherited]`

Returns the `size()` of the largest possible vector.

Definition at line 650 of file `stl_vector.h`.

**4.490.2.33** reference `std::vector<_State, std::allocator<_State>>::operator[] ( size_type __n )` `[inline]`,  
`[inherited]`

Subscript access to the data contained in the vector.

#### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

#### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 770 of file `stl_vector.h`.

**4.490.2.34** const\_reference `std::vector<_State, std::allocator<_State>>::operator[] ( size_type __n ) const`  
`[inline]`, `[inherited]`

Subscript access to the data contained in the vector.

#### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 785 of file `stl_vector.h`.

**4.490.2.35** void `std::vector<_State, std::allocator<_State>>::pop_back ( )` `[inline]`, `[inherited]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 937 of file `stl_vector.h`.

**4.490.2.36** void `std::vector<_State, std::allocator<_State>>::push_back ( const value_type & __x )` `[inline]`,  
`[inherited]`

Add data to the end of the vector.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 901 of file stl\_vector.h.

**4.490.2.37** `reverse_iterator std::vector<_State, std::allocator<_State>>::rbegin ( ) [inline],  
[noexcept], [inherited]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 574 of file stl\_vector.h.

**4.490.2.38** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::rbegin ( ) const [inline],  
[noexcept], [inherited]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file stl\_vector.h.

**4.490.2.39** `reverse_iterator std::vector<_State, std::allocator<_State>>::rend ( ) [inline], [noexcept],  
[inherited]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file stl\_vector.h.

**4.490.2.40** `const_reverse_iterator std::vector<_State, std::allocator<_State>>::rend ( ) const [inline],  
[noexcept], [inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file stl\_vector.h.

**4.490.2.41** `void std::vector<_State, std::allocator<_State>>::reserve ( size_type __n ) [inherited]`

Attempt to preallocate enough memory for specified number of elements.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

#### Exceptions

|                                |                                               |
|--------------------------------|-----------------------------------------------|
| <code>std::length_error</code> | If <i>n</i> exceeds <code>max_size()</code> . |
|--------------------------------|-----------------------------------------------|

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

**4.490.2.42** `void std::vector<_State, std::allocator<_State>>::resize ( size_type __new_size ) [inline],  
[inherited]`

Resizes the vector to the specified number of elements.

## Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain. |
|-------------------------|-----------------------------------------------|

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 664 of file `stl_vector.h`.

**4.490.2.43** `void std::vector<_State, std::allocator<_State>>::resize ( size_type __new_size, const value_type & __x )`  
`[inline], [inherited]`

Resizes the vector to the specified number of elements.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain.     |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 684 of file `stl_vector.h`.

**4.490.2.44** `void std::vector<_State, std::allocator<_State>>::shrink_to_fit ( )` `[inline], [inherited]`

A non-binding request to reduce capacity() to size().

Definition at line 716 of file `stl_vector.h`.

**4.490.2.45** `size_type std::vector<_State, std::allocator<_State>>::size ( ) const` `[inline], [noexcept], [inherited]`

Returns the number of elements in the vector.

Definition at line 645 of file `stl_vector.h`.

**4.490.2.46** `void std::vector<_State, std::allocator<_State>>::swap ( vector<_State> & __x )` `[inline], [noexcept], [inherited]`

Swaps data with another vector.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A vector of the same element and allocator types. |
|------------------|---------------------------------------------------|

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

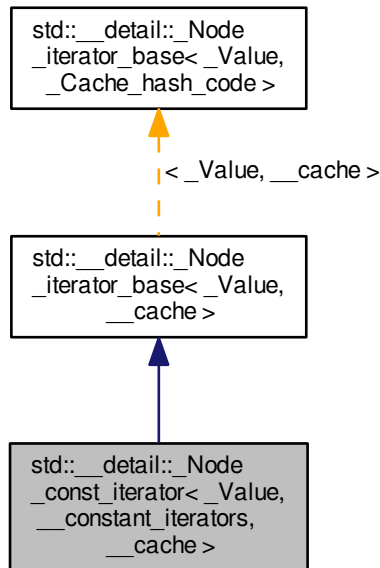
Definition at line 1108 of file `stl_vector.h`.

The documentation for this class was generated from the following file:

- [regex\\_nfa.h](#)

#### 4.491 `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

#### Public Member Functions

- **\_Node\_const\_iterator** (`__node_type *__p`)
- **\_Node\_const\_iterator** (`const \_Node\_iterator< _Value, __constant_iterators, __cache > &__x`)
- `void _M_incr ()`
- reference **operator\*** () const
- [\\_Node\\_const\\_iterator](#) & **operator++** ()
- [\\_Node\\_const\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

#### Public Attributes

- `__node_type * _M_cur`

## 4.491.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >
```

Node const\_iterators, used to iterate through all the hashtable.

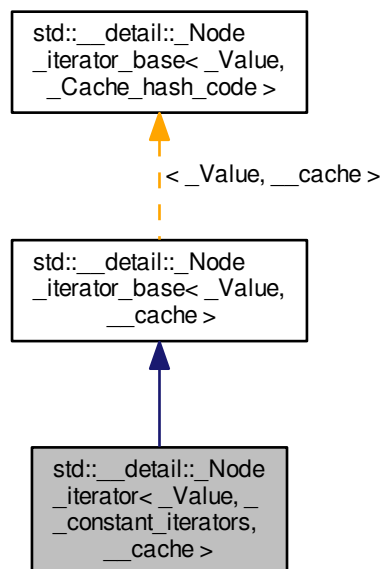
Definition at line 282 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.492 std::\_\_detail::\_Node\_iterator&lt; \_Value, \_\_constant\_iterators, \_\_cache &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >:



## Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- using **pointer** = typename `std::conditional< __constant_iterators, const _Value *, _Value * >::type`
- using **reference** = typename `std::conditional< __constant_iterators, const _Value &, _Value & >::type`
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Node\_iterator** (\_\_node\_type \*\_\_p)
- void **\_M\_incr** ()
- reference **operator\*** () const
- [\\_Node\\_iterator](#) & **operator++** ()
- [\\_Node\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

## Public Attributes

- \_\_node\_type \* **\_M\_cur**

## 4.492.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache> struct std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >
```

Node iterators, used to iterate through all the hashtable.

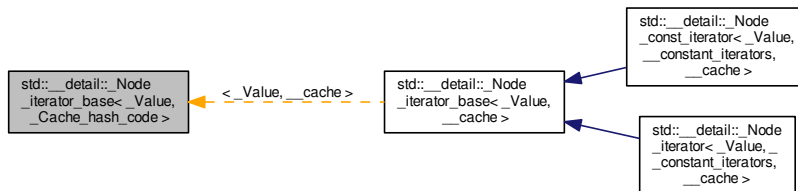
Definition at line 231 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.493 std::\_\_detail::\_Node\_iterator\_base&lt; \_Value, \_Cache\_hash\_code &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Node\_iterator\_base< \_Value, \_Cache\_hash\_code >:



## Public Types

- using **\_\_node\_type** = [\\_Hash\\_node](#)< \_Value, \_Cache\_hash\_code >

## Public Member Functions

- **\_Node\_iterator\_base** ([\\_\\_node\\_type](#) \*\_\_p)
- void **\_M\_incr** ()

## Public Attributes

- [\\_\\_node\\_type](#) \* **\_M\_cur**

## 4.493.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>struct std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >
```

Base class for node iterators.

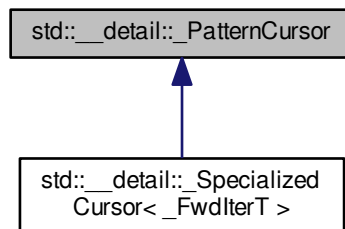
Definition at line 203 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.494 std::\_\_detail::\_PatternCursor Struct Reference

Inheritance diagram for std::\_\_detail::\_PatternCursor:



## Public Member Functions

- virtual bool **\_M\_at\_end** () const =0
- virtual void **\_M\_next** ()=0

## 4.494.1 Detailed Description

ABC for pattern matching.

Definition at line 44 of file regex\_cursor.h.

The documentation for this struct was generated from the following file:

- [regex\\_cursor.h](#)

## 4.495 std::\_\_detail::\_Prime\_rehash\_policy Struct Reference

## Public Types

- enum { **\_S\_n\_primes** }
- typedef std::size\_t **\_State**

## Public Member Functions

- **\_Prime\_rehash\_policy** (float \_\_z=1.0)
- std::size\_t **\_M\_bkt\_for\_elements** (std::size\_t \_\_n) const
- [std::pair](#)< bool, std::size\_t > **\_M\_need\_rehash** (std::size\_t \_\_n\_bkt, std::size\_t \_\_n\_elt, std::size\_t \_\_n\_ins) const
- std::size\_t **\_M\_next\_bkt** (std::size\_t \_\_n) const
- void **\_M\_reset** (\_State \_\_state)
- \_State **\_M\_state** () const
- float **max\_load\_factor** () const noexcept

## Public Attributes

- float **\_M\_max\_load\_factor**
- std::size\_t **\_M\_next\_resize**

## Static Public Attributes

- static const std::size\_t **\_S\_growth\_factor**

## 4.495.1 Detailed Description

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

Definition at line 357 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.496 std::\_\_detail::\_RangeMatcher&lt; \_InIterT, \_TraitsT &gt; Struct Template Reference

## Public Types

- typedef \_TraitsT::char\_type **\_CharT**
- typedef [std::basic\\_string](#)< \_CharT > **\_StringT**

## Public Member Functions

- **\_RangeMatcher** (bool \_\_is\_non\_matching, const \_TraitsT &\_\_t=\_TraitsT())
- void **\_M\_add\_char** (\_CharT \_\_c)
- void **\_M\_add\_character\_class** (const [\\_StringT](#) &\_\_s)
- void **\_M\_add\_collating\_element** (const [\\_StringT](#) &\_\_s)

- void `_M_add_equivalence_class` (const [\\_StringT](#) &\_\_s)
- void `_M_make_range` ()
- bool `operator()` (const [\\_PatternCursor](#) &\_\_pc) const

#### Public Attributes

- bool `_M_is_non_matching`
- const `_TraitsT` & `_M_traits`

#### 4.496.1 Detailed Description

template<typename `_InIterT`, typename `_TraitsT`>struct std::\_\_detail::RangeMatcher< `_InIterT`, `_TraitsT` >

Matches a character range (bracket expression)

Definition at line 152 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

#### 4.497 `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >` Struct Template Reference

#### Public Types

- using `__hashtable` = [\\_Hashtable](#)< `_Key`, `_Value`, `_Alloc`, `_ExtractKey`, `_Equal`, `_H1`, `_H2`, `_Hash`, [\\_Prime\\_rehash\\_policy](#), `_Traits` >

#### Public Member Functions

- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_z)
- void `reserve` (std::size\_t \_\_n)

#### 4.497.1 Detailed Description

template<typename `_Key`, typename `_Value`, typename `_Alloc`, typename `_ExtractKey`, typename `_Equal`, typename `_H1`, typename `_H2`, typename `_Hash`, typename `_Traits`>struct std::\_\_detail::Rehash\_base< `_Key`, `_Value`, `_Alloc`, `_ExtractKey`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_Prime_rehash_policy`, `_Traits` >

Specialization.

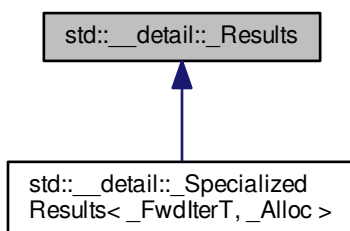
Definition at line 794 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.498 std::\_\_detail::\_Results Struct Reference

Inheritance diagram for std::\_\_detail::\_Results:

**Public Member Functions**

- virtual void **\_M\_set\_matched** (int \_\_i, bool \_\_is\_matched)=0
- virtual void **\_M\_set\_pos** (int \_\_i, int \_\_j, const [\\_PatternCursor](#) &\_\_p)=0

## 4.498.1 Detailed Description

Provides a generic facade for a templated match\_results.

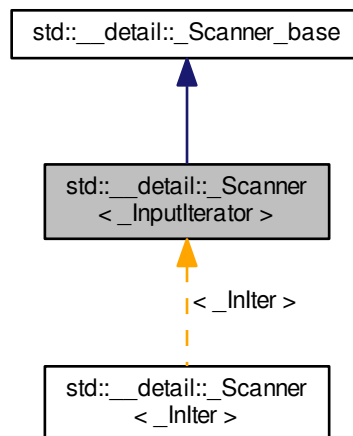
Definition at line 77 of file regex\_nfa.h.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

## 4.499 std::\_\_detail::\_Scanner&lt; \_InputIterator &gt; Class Template Reference

Inheritance diagram for std::\_\_detail::\_Scanner< \_InputIterator >:



## Public Types

- typedef std::iterator\_traits< \_IteratorT >::value\_type **\_CharT**
- typedef const [std::ctype](#)< \_CharT > **\_CtypeT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef \_InputIterator **\_IteratorT**
- typedef unsigned int **\_StateT**
- typedef [std::basic\\_string](#)< \_CharT > **\_StringT**
- enum **\_TokenT** {  
     \_S\_token\_anychar, \_S\_token\_backref, \_S\_token\_bracket\_begin, \_S\_token\_bracket\_end,  
     \_S\_token\_inverse\_class, \_S\_token\_char\_class\_name, \_S\_token\_closure0, \_S\_token\_closure1,  
     \_S\_token\_collelem\_multi, \_S\_token\_collelem\_single, \_S\_token\_collsymbol, \_S\_token\_comma,  
     \_S\_token\_dash, \_S\_token\_dup\_count, \_S\_token\_eof, \_S\_token\_equiv\_class\_name,  
     \_S\_token\_interval\_begin, \_S\_token\_interval\_end, \_S\_token\_line\_begin, \_S\_token\_line\_end,  
     \_S\_token\_opt, \_S\_token\_or, \_S\_token\_ord\_char, \_S\_token\_quoted\_char,  
     \_S\_token\_subexpr\_begin, \_S\_token\_subexpr\_end, \_S\_token\_word\_begin, \_S\_token\_word\_end,  
     \_S\_token\_unknown }

## Public Member Functions

- **\_Scanner** (\_IteratorT \_\_begin, \_IteratorT \_\_end, \_FlagT \_\_flags, [std::locale](#) \_\_loc)
- void **\_M\_advance** ()
- **\_TokenT** **\_M\_token** () const
- const **\_StringT** & **\_M\_value** () const

#### Static Public Attributes

- static constexpr \_StateT **\_S\_state\_at\_start**
- static constexpr \_StateT **\_S\_state\_in\_brace**
- static constexpr \_StateT **\_S\_state\_in\_bracket**

#### 4.499.1 Detailed Description

template<typename \_InputIterator> class std::\_\_detail::\_Scanner<\_InputIterator >

struct \_Scanner. Scans an input range for regex tokens.

The \_Scanner class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

Definition at line 65 of file regex\_compiler.h.

#### 4.499.2 Member Enumeration Documentation

4.499.2.1 template<typename \_InputIterator> enum std::\_\_detail::\_Scanner::\_TokenT

Token types returned from the scanner.

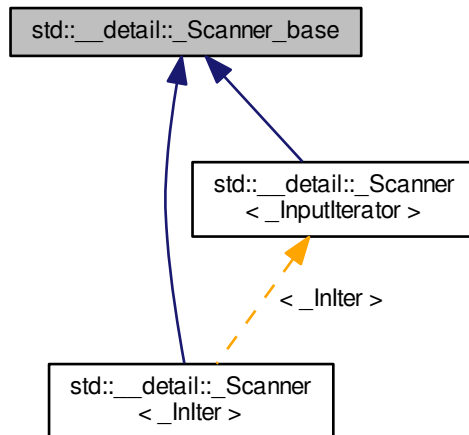
Definition at line 75 of file regex\_compiler.h.

The documentation for this class was generated from the following file:

- [regex\\_compiler.h](#)

## 4.500 std::\_\_detail::\_Scanner\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Scanner\_base:



## Public Types

- typedef unsigned int **\_StateT**

## Static Public Attributes

- static constexpr \_StateT **\_S\_state\_at\_start**
- static constexpr \_StateT **\_S\_state\_in\_brace**
- static constexpr \_StateT **\_S\_state\_in\_bracket**

## 4.500.1 Detailed Description

Base class for scanner.

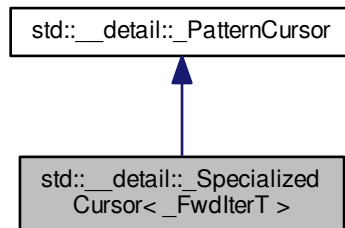
Definition at line 43 of file `regex_compiler.h`.

The documentation for this struct was generated from the following file:

- [regex\\_compiler.h](#)

## 4.501 std::\_\_detail::\_SpecializedCursor&lt; \_FwdIterT &gt; Class Template Reference

Inheritance diagram for std::\_\_detail::\_SpecializedCursor< \_FwdIterT >:



## Public Member Functions

- **\_SpecializedCursor** (const \_FwdIterT &\_\_b, const \_FwdIterT \_\_e)
- bool **\_M\_at\_end** () const
- const \_FwdIterT & **\_M\_begin** () const
- std::iterator\_traits< \_FwdIterT >::value\_type **\_M\_current** () const
- const \_FwdIterT & **\_M\_end** () const
- void **\_M\_next** ()
- \_FwdIterT **\_M\_pos** () const

## 4.501.1 Detailed Description

```
template<typename _FwdIterT>class std::__detail::_SpecializedCursor< _FwdIterT >
```

Provides a cursor into the specific target string.

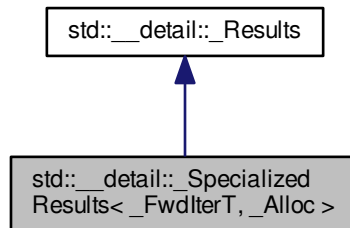
Definition at line 53 of file regex\_cursor.h.

The documentation for this class was generated from the following file:

- [regex\\_cursor.h](#)

## 4.502 std::\_\_detail::\_SpecializedResults&lt; \_FwdIterT, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_\_detail::\_SpecializedResults< \_FwdIterT, \_Alloc >:



## Public Member Functions

- **\_SpecializedResults** (const \_Automaton::\_SizeT \_\_size, const [\\_SpecializedCursor](#)< \_FwdIterT > &\_\_cursor, [match\\_results](#)< \_FwdIterT, \_Alloc > &\_\_m)
- void **\_M\_set\_matched** (int \_\_i, bool \_\_is\_matched)
- void **\_M\_set\_pos** (int \_\_i, int \_\_j, const [\\_PatternCursor](#) &\_\_pc)

## 4.502.1 Detailed Description

```
template<typename _FwdIterT, typename _Alloc>class std::__detail::_SpecializedResults< _FwdIterT, _Alloc >
```

A \_Results facade specialized for wrapping a templated match\_results.

Definition at line 55 of file `regex_grep_matcher.h`.

The documentation for this class was generated from the following file:

- [regex\\_grep\\_matcher.h](#)

## 4.503 std::\_\_detail::\_StartTagger&lt; \_FwdIterT, \_TraitsT &gt; Struct Template Reference

## Public Member Functions

- **\_StartTagger** (int \_\_i)
- void **operator()** (const [\\_PatternCursor](#) &\_\_pc, [\\_Results](#) &\_\_r)

## Public Attributes

- int **\_M\_index**

## 4.503.1 Detailed Description

```
template<typename _FwdIterT, typename _TraitsT> struct std::__detail::_StartTagger< _FwdIterT, _TraitsT >
```

Start state tag.

Definition at line 88 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

## 4.504 std::\_\_detail::\_State Struct Reference

## Public Types

- typedef int **\_OpcodeT**

## Public Member Functions

- **\_State** (**\_OpcodeT** \_\_opcode)
- **\_State** (const [\\_Matcher](#) &\_\_m)
- **\_State** (**\_OpcodeT** \_\_opcode, unsigned int \_\_s, const [\\_Tagger](#) &\_\_t)
- **\_State** ([\\_StateIdT](#) \_\_next, [\\_StateIdT](#) \_\_alt)

## Public Attributes

- [\\_StateIdT](#) **\_M\_alt**
- [\\_Matcher](#) **\_M\_matches**
- [\\_StateIdT](#) **\_M\_next**
- **\_OpcodeT** **\_M\_opcode**
- unsigned int **\_M\_subexpr**
- [\\_Tagger](#) **\_M\_tagger**

## 4.504.1 Detailed Description

```
struct _State
```

An individual state in an NFA

In this case a "state" is an entry in the NFA definition coupled with its outgoing transition(s). All states have a single outgoing transition, except for accepting states (which have no outgoing transitions) and alt states, which have two outgoing transitions.

Definition at line 211 of file `regex_nfa.h`.

The documentation for this struct was generated from the following file:

- [regex\\_nfa.h](#)

## 4.505 std::\_\_detail::\_StateSeq Class Reference

## Public Member Functions

- [\\_StateSeq](#) ([\\_Nfa](#) &\_\_ss, [\\_StateIdT](#) \_\_s, [\\_StateIdT](#) \_\_e=[\\_S\\_invalid\\_state\\_id](#))
- [\\_StateSeq](#) (const [\\_StateSeq](#) &\_\_e1, const [\\_StateSeq](#) &\_\_e2)
- [\\_StateSeq](#) (const [\\_StateSeq](#) &\_\_e, [\\_StateIdT](#) \_\_id)
- [\\_StateSeq](#) (const [\\_StateSeq](#) &\_\_rhs)
- void [\\_M\\_append](#) ([\\_StateIdT](#) \_\_id)
- void [\\_M\\_append](#) ([\\_StateSeq](#) &\_\_rhs)
- [\\_StateIdT](#) [\\_M\\_clone](#) ()
- [\\_StateIdT](#) [\\_M\\_front](#) () const
- void [\\_M\\_push\\_back](#) ([\\_StateIdT](#) \_\_id)
- [\\_StateSeq](#) & [operator=](#) (const [\\_StateSeq](#) &\_\_rhs)

## 4.505.1 Detailed Description

Describes a sequence of one or more [\\_State](#), its current start and end(s). This structure contains fragments of an NFA during construction.

Definition at line 352 of file [regex\\_nfa.h](#).

The documentation for this class was generated from the following files:

- [regex\\_nfa.h](#)
- [regex\\_nfa.tcc](#)

## 4.506 std::\_\_exception\_ptr::exception\_ptr Class Reference

## Public Member Functions

- [exception\\_ptr](#) (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) (nullptr\_t) noexcept
- [exception\\_ptr](#) ([exception\\_ptr](#) &&\_\_o) noexcept
- class [type\\_info](#) \* [\\_\\_cxa\\_exception\\_type](#) () const noexcept [\\_\\_attribute\\_\\_\(\(\\_\\_pure\\_\\_\)\)](#)
- [operator bool](#) () const
- [exception\\_ptr](#) & [operator=](#) (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) & [operator=](#) ([exception\\_ptr](#) &&\_\_o) noexcept
- void [swap](#) ([exception\\_ptr](#) &) noexcept

## Friends

- bool [operator==](#) (const [exception\\_ptr](#) &, const [exception\\_ptr](#) &) noexcept [\\_\\_attribute\\_\\_\(\(\\_\\_pure\\_\\_\)\)](#)
- [exception\\_ptr](#) [std::current\\_exception](#) () noexcept
- void [std::rethrow\\_exception](#) ([exception\\_ptr](#))

## 4.506.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 73 of file exception\_ptr.h.

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

## 4.507 std::\_\_has\_iterator\_category\_helper&lt; \_Tp &gt; Class Template Reference

Inherits std::\_\_sfn\_oe\_types.

## Static Public Attributes

- static constexpr bool **value**

## Private Types

- typedef char **\_\_one**

## 4.507.1 Detailed Description

```
template<typename _Tp>class std::__has_iterator_category_helper< _Tp >
```

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the Iterator argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

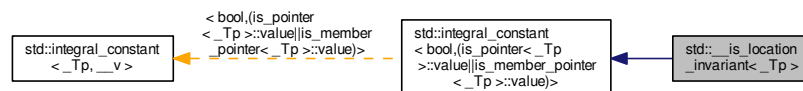
Definition at line 142 of file stl\_iterator\_base\_types.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.508 std::\_\_is\_location\_invariant&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::\_\_is\_location\_invariant< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)  
< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.508.1 Detailed Description

template<typename \_Tp>struct std::\_\_is\_location\_invariant< \_Tp >

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Also implies a trivial copy constructor and assignment operator.

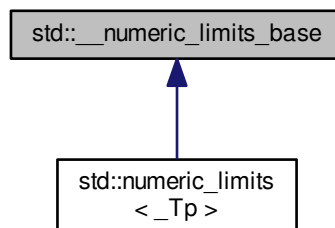
Definition at line 1781 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.509 std::\_\_numeric\_limits\_base Struct Reference

Inheritance diagram for std::\_\_numeric\_limits\_base:



## Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)

- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

#### 4.509.1 Detailed Description

Part of `std::numeric_limits`.

The `static const` members are usable as integral constant expressions.

##### Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the `std::numeric_limits` class.

Definition at line 191 of file `limits`.

#### 4.509.2 Member Data Documentation

##### 4.509.2.1 constexpr int std::\_\_numeric\_limits\_base::digits [static]

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 200 of file `limits`.

##### 4.509.2.2 constexpr int std::\_\_numeric\_limits\_base::digits10 [static]

The number of base 10 digits that can be represented without change.

Definition at line 203 of file `limits`.

##### 4.509.2.3 constexpr float\_denorm\_style std::\_\_numeric\_limits\_base::has\_denorm [static]

See `std::float_denorm_style` for more information.

Definition at line 255 of file `limits`.

**4.509.2.4 constexpr bool std::\_\_numeric\_limits\_base::has\_denorm\_loss** [static]

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 259 of file limits.

**4.509.2.5 constexpr bool std::\_\_numeric\_limits\_base::has\_infinity** [static]

True if the type has a representation for positive infinity.

Definition at line 244 of file limits.

**4.509.2.6 constexpr bool std::\_\_numeric\_limits\_base::has\_quiet\_NaN** [static]

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 248 of file limits.

**4.509.2.7 constexpr bool std::\_\_numeric\_limits\_base::has\_signaling\_NaN** [static]

True if the type has a representation for a signaling Not a Number.

Definition at line 252 of file limits.

**4.509.2.8 constexpr bool std::\_\_numeric\_limits\_base::is\_bounded** [static]

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 268 of file limits.

**4.509.2.9 constexpr bool std::\_\_numeric\_limits\_base::is\_exact** [static]

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 220 of file limits.

**4.509.2.10 constexpr bool std::\_\_numeric\_limits\_base::is\_iec559** [static]

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 263 of file limits.

**4.509.2.11 constexpr bool std::\_\_numeric\_limits\_base::is\_integer** [static]

True if the type is integer.

Definition at line 215 of file limits.

**4.509.2.12 constexpr bool std::\_\_numeric\_limits\_base::is\_modulo** [static]

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 277 of file limits.

**4.509.2.13** constexpr bool std::\_\_numeric\_limits\_base::is\_signed [static]

True if the type is signed.

Definition at line 212 of file limits.

**4.509.2.14** constexpr bool std::\_\_numeric\_limits\_base::is\_specialized [static]

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 195 of file limits.

**4.509.2.15** constexpr int std::\_\_numeric\_limits\_base::max\_digits10 [static]

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 208 of file limits.

**4.509.2.16** constexpr int std::\_\_numeric\_limits\_base::max\_exponent [static]

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 237 of file limits.

**4.509.2.17** constexpr int std::\_\_numeric\_limits\_base::max\_exponent10 [static]

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 241 of file limits.

**4.509.2.18** constexpr int std::\_\_numeric\_limits\_base::min\_exponent [static]

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 228 of file limits.

**4.509.2.19** constexpr int std::\_\_numeric\_limits\_base::min\_exponent10 [static]

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 232 of file limits.

**4.509.2.20** constexpr int std::\_\_numeric\_limits\_base::radix [static]

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 224 of file limits.

**4.509.2.21** constexpr float\_round\_style std::\_\_numeric\_limits\_base::round\_style [static]

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 288 of file limits.

4.509.2.22 `constexpr bool std::__numeric_limits_base::tinyness_before` `[static]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 283 of file `limits`.

4.509.2.23 `constexpr bool std::__numeric_limits_base::traps` `[static]`

True if trapping is implemented for this type.

Definition at line 280 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.510 `std::__parallel::CRandNumber<_MustBeInt>` Struct Template Reference

### Public Member Functions

- `int operator() (int __limit)`

#### 4.510.1 Detailed Description

`template<typename _MustBeInt = int>struct std::__parallel::CRandNumber<_MustBeInt>`

Functor wrapper for `std::rand()`.

Definition at line 1656 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

## 4.511 `std::__profile::bitset<_Nb>` Class Template Reference

Inherits `bitset<_Nb>`.

### Public Member Functions

- `constexpr bitset (unsigned long long __val) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`bitset (const std::basic_string<_CharT, _Traits, _Alloc> &__str, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __pos=0, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __n=(std::basic_string<_CharT, _Traits, _Alloc>::npos))`
- `template<class _CharT, class _Traits, class _Alloc>`  
`bitset (const std::basic_string<_CharT, _Traits, _Alloc> &__str, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __pos, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __n, _CharT __zero, _CharT __one= _CharT('1'))`
- `bitset (const _Base &__x)`
- `template<typename _CharT>`  
`bitset (const _CharT *__str, typename std::basic_string<_CharT>::size_type __n=std::basic_string<_CharT>::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))`

- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- [bitset](#)<\_Nb> & [flip](#) () noexcept
- [bitset](#)<\_Nb> & [flip](#) (size\_t \_\_pos)
- bool [operator!=](#) (const [bitset](#)<\_Nb> &\_\_rhs) const noexcept
- [bitset](#)<\_Nb> & [operator&=](#) (const [bitset](#)<\_Nb> &\_\_rhs) noexcept
- [bitset](#)<\_Nb> [operator<<](#) (size\_t \_\_pos) const noexcept
- [bitset](#)<\_Nb> & [operator<<=](#) (size\_t \_\_pos) noexcept
- bool [operator==](#) (const [bitset](#)<\_Nb> &\_\_rhs) const noexcept
- [bitset](#)<\_Nb> [operator>>](#) (size\_t \_\_pos) const noexcept
- [bitset](#)<\_Nb> & [operator>>=](#) (size\_t \_\_pos) noexcept
- reference [operator\[\]](#) (size\_t \_\_pos)
- constexpr bool [operator\[\]](#) (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & [operator^=](#) (const [bitset](#)<\_Nb> &\_\_rhs) noexcept
- [bitset](#)<\_Nb> & [operator|=](#) (const [bitset](#)<\_Nb> &\_\_rhs) noexcept
- [bitset](#)<\_Nb> [operator~](#) () const noexcept
- [bitset](#)<\_Nb> & [reset](#) () noexcept
- [bitset](#)<\_Nb> & [reset](#) (size\_t \_\_pos)
- [bitset](#)<\_Nb> & [set](#) () noexcept
- [bitset](#)<\_Nb> & [set](#) (size\_t \_\_pos, bool \_\_val=true)
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
[std::basic\\_string](#)<\_CharT,  
\_Traits, \_Alloc> [to\\_string](#) () const
- template<class \_CharT, class \_Traits, class \_Alloc>  
[std::basic\\_string](#)<\_CharT,  
\_Traits, \_Alloc> [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT, typename \_Traits>  
[std::basic\\_string](#)<\_CharT,  
\_Traits, [std::allocator](#)  
<\_CharT>> [to\\_string](#) () const
- template<class \_CharT, class \_Traits>  
[std::basic\\_string](#)<\_CharT,  
\_Traits, [std::allocator](#)  
<\_CharT>> [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT>  
[std::basic\\_string](#)<\_CharT,  
[std::char\\_traits](#)<\_CharT>  
, [std::allocator](#)<\_CharT>> [to\\_string](#) () const
- template<class \_CharT>  
[std::basic\\_string](#)<\_CharT,  
[std::char\\_traits](#)<\_CharT>  
, [std::allocator](#)<\_CharT>> [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- [std::basic\\_string](#)<char,  
[std::char\\_traits](#)<char>  
, [std::allocator](#)<char>> [to\\_string](#) () const
- [std::basic\\_string](#)<char,  
[std::char\\_traits](#)<char>  
, [std::allocator](#)<char>> [to\\_string](#) (char \_\_zero, char \_\_one='1') const

## 4.511.1 Detailed Description

```
template<size_t _Nb>class std::__profile::bitset< _Nb >
```

Class `std::bitset` wrapper with performance instrumentation.

Definition at line 40 of file `profile/bitset`.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

4.512 `std::__profile::deque< _Tp, _Allocator >` Class Template Reference

Inherits `deque< _Tp, _Allocator >`.

## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **deque** (const `_Allocator` &\_\_a=\_Allocator())
- **deque** (size\_type \_\_n)
- **deque** (size\_type \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` , typename = std::\_RequireInputIter<\_InputIterator>>  
  **deque** (\_InputIterator \_\_first, \_InputIterator \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- **deque** (const [deque](#) &\_\_x)
- **deque** (const [\\_Base](#) &\_\_x)
- **deque** ([deque](#) &&\_\_x)
- **deque** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- [\\_Base](#) & **\_M\_base** () noexcept
- const [\\_Base](#) & **\_M\_base** () const noexcept
- template<typename `_InputIterator` , typename = std::\_RequireInputIter<\_InputIterator>>  
  void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_t)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- reference **back** ()
- const\_reference **back** () const

- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (iterator \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void **emplace\_back** (\_Args &&...\_\_args)
- template<typename... \_Args>  
void **emplace\_front** (\_Args &&...\_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- iterator **insert** (iterator \_\_position, const \_Tp &\_\_x)
- iterator **insert** (iterator \_\_position, \_Tp &&\_\_x)
- void **insert** (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- void **insert** (iterator \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- deque & **operator=** (const deque &\_\_x)
- deque & **operator=** (deque &&\_\_x)
- deque & **operator=** (initializer\_list< value\_type > \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (const \_Tp &\_\_x)
- void **push\_back** (\_Tp &&\_\_x)
- void **push\_front** (const \_Tp &\_\_x)
- void **push\_front** (\_Tp &&\_\_x)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- void **swap** (deque &\_\_x)

## 4.512.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__profile::deque< _Tp, _Allocator >
```

Class `std::deque` wrapper with performance instrumentation.

Definition at line 40 of file `profile/deque`.

The documentation for this class was generated from the following file:

- [profile/deque](#)

4.513 `std::__profile::forward_list< _Tp, _Alloc >` Class Template Reference

Inherits `forward_list< _Tp, _Alloc >`.

## Public Types

- typedef `_Base::size_type` **size\_type**

## Public Member Functions

- **forward\_list** (const `_Alloc` &\_\_al=`_Alloc`())
- **forward\_list** (const [forward\\_list](#) &\_\_list, const `_Alloc` &\_\_al)
- **forward\_list** ([forward\\_list](#) &&\_\_list, const `_Alloc` &\_\_al)
- **forward\_list** (size\_type \_\_n, const `_Alloc` &\_\_al=`_Alloc`())
- **forward\_list** (size\_type \_\_n, const `_Tp` &\_\_value, const `_Alloc` &\_\_al=`_Alloc`())
- template<typename `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`  
**forward\_list** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Alloc` &\_\_al=`_Alloc`())
- **forward\_list** (const [forward\\_list](#) &\_\_list)
- **forward\_list** ([forward\\_list](#) &&\_\_list) noexcept
- **forward\_list** ([std::initializer\\_list](#)< `_Tp` > \_\_il, const `_Alloc` &\_\_al=`_Alloc`())
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- [forward\\_list](#) & **operator=** (const [forward\\_list](#) &\_\_list)
- [forward\\_list](#) & **operator=** ([forward\\_list](#) &&\_\_list) noexcept([\\_Node\\_alloc\\_traits](#)
- [forward\\_list](#) & **operator=** ([std::initializer\\_list](#)< `_Tp` > \_\_il)

## 4.513.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>class std::__profile::forward_list< _Tp, _Alloc >
```

Class `std::forward_list` wrapper with performance instrumentation.

Definition at line 44 of file `profile/forward_list`.

The documentation for this class was generated from the following file:

- [profile/forward\\_list](#)

4.514 `std::__profile::list< _Tp, _Allocator >` Class Template Reference

Inherits `list< _Tp, _Allocator >`.

## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__iterator_tracker`  
    < typename  
        \_Base::const\_iterator, `list` > **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator`  
    < const\_iterator > **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__iterator_tracker`  
    < typename \_Base::iterator,  
        `list` > **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator`  
    < iterator > **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **list** (const `_Allocator` &\_\_a=\_Allocator())
- **list** (size\_type \_\_n)
- **list** (size\_type \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`  
    **list** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- **list** (const `list` &\_\_x)
- **list** (const `_Base` &\_\_x)
- **list** (`list` &&\_\_x) noexcept
- **list** (`initializer_list`< value\_type > \_\_l, const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_profile\_find** () const
- void **\_M\_profile\_iterate** (int \_\_rewind=0) const
- void **assign** (`initializer_list`< value\_type > \_\_l)
- template<typename `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`  
    void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_t)
- reference **back** ()
- const\_reference **back** () const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept

- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (iterator \_\_position, \_Args &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_position, iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- iterator **insert** (iterator \_\_position, const \_Tp & \_\_x)
- iterator **insert** (iterator \_\_position, \_Tp && \_\_x)
- void **insert** (iterator \_\_position, [initializer\\_list](#)< value\_type > \_\_l)
- void **insert** (iterator \_\_position, size\_type \_\_n, const \_Tp & \_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **merge** ([list](#) && \_\_x)
- void **merge** ([list](#) & \_\_x)
- template<class \_Compare >  
void **merge** ([list](#) && \_\_x, \_Compare \_\_comp)
- template<typename \_Compare >  
void **merge** ([list](#) & \_\_x, \_Compare \_\_comp)
- [list](#) & **operator=** (const [list](#) & \_\_x)
- [list](#) & **operator=** ([list](#) && \_\_x)
- [list](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_front** (const value\_type & \_\_x)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- void **remove** (const \_Tp & \_\_value)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp & \_\_c)
- void **sort** ()
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **splice** (iterator \_\_position, [list](#) && \_\_x)
- void **splice** (iterator \_\_position, [list](#) & \_\_x)
- void **splice** (iterator \_\_position, [list](#) & \_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, [list](#) && \_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, [list](#) && \_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_position, [list](#) & \_\_x, iterator \_\_first, iterator \_\_last)
- void **swap** ([list](#) & \_\_x)
- void **unique** ()
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_binary\_pred)

## 4.514.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__profile::list<_Tp, _Allocator>
```

List wrapper with performance instrumentation.

Definition at line 42 of file `profile/list`.

The documentation for this class was generated from the following file:

- [profile/list](#)

4.515 `std::__profile::map<_Key, _Tp, _Compare, _Allocator>` Class Template Reference

Inherits `map<_Key, _Tp, _Compare, _Allocator>`.

## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef [std::reverse\\_iterator](#)  
    `< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef [std::reverse\\_iterator](#)  
    `< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef [std::pair](#)`< const _Key,`  
    `_Tp >` **value\_type**

## Public Member Functions

- **map** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
    **map** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **map** (`const map &__x`)
- **map** (`const _Base &__x`)
- **map** (`map &&__x`) `noexcept(is_nothrow_copy_constructible<_Compare>)`
- **map** (`initializer_list< value_type > __l, const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `_Base & _M_base () noexcept`
- `const _Base & _M_base () const noexcept`
- `mapped_type & at (const key_type &__k)`

- `const mapped_type & at (const key_type &__k) const`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear () noexcept`
- `size_type count (const key_type &__x) const`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args>`  
`std::pair< iterator, bool > emplace (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __pos, _Args &&... __args)`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator,`  
`const_iterator > equal_range (const key_type &__x) const`
- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`
- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`std::pair< iterator, bool > insert (_Pair &&__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert (const_iterator __position, _Pair &&__x)`
- `template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `map & operator= (const map &__x)`
- `map & operator= (map &&__x)`
- `map & operator= (initializer_list< value_type > __l)`
- `mapped_type & operator[] (const key_type &__k)`
- `mapped_type & operator[] (key_type &&__k)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void swap (map &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

## 4.515.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std-
::pair<const _Key, _Tp>>> class std::_profile::map<_Key, _Tp, _Compare, _Allocator>
```

Class `std::map` wrapper with performance instrumentation.

Definition at line 41 of file `profile/map.h`.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

4.516 `std::_profile::multimap<_Key, _Tp, _Compare, _Allocator>` Class Template Reference

Inherits `multimap<_Key, _Tp, _Compare, _Allocator>`.

## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp>` **value\_type**

## Public Member Functions

- **multimap** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
**multimap** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multimap** (`const multimap &__x`)
- **multimap** (`const \_Base &__x`)
- **multimap** (`multimap &&__x`) `noexcept(is_nothrow_copy_constructible<_Compare>)`
- **multimap** (`initializer\_list< value\_type> __l, const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `\_Base & \_M\_base () noexcept`
- `const \_Base & \_M\_base () const noexcept`
- `iterator begin () noexcept`

- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- iterator **erase** (iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- iterator **insert** (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (\_Pair &&\_\_x)
- void **insert** (std::initializer\_list< value\_type > \_\_list)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- multimap & **operator=** (const multimap &\_\_x)
- multimap & **operator=** (multimap &&\_\_x)
- multimap & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** (multimap &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 4.516.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>> class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class std::multimap wrapper with performance instrumentation.

Definition at line 41 of file profile/multimap.h.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

## 4.517 `std::_profile::multiset<_Key, _Compare, _Allocator>` Class Template Reference

Inherits `multiset<_Key, _Compare, _Allocator>`.

### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

### Public Member Functions

- **multiset** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
**multiset** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multiset** (`const multiset &__x`)
- **multiset** (`const _Base &__x`)
- **multiset** (`multiset &&__x`) `noexcept(is_nothrow_copy_constructible<_Compare>)`
- **multiset** (`initializer_list<value_type> __l, const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `_Base &_M_base () noexcept`
- `const _Base &_M_base () const noexcept`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear () noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args>`  
`iterator emplace (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __pos, _Args &&... __args)`
- `iterator end () noexcept`

- `const_iterator` **end** () `const` `noexcept`
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type &__x`)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type &__x`) `const`
- `iterator` **erase** (`const_iterator __position`)
- `size_type` **erase** (`const key_type &__x`)
- `iterator` **erase** (`const_iterator __first, const_iterator __last`)
- `iterator` **find** (`const key_type &__x`)
- `const_iterator` **find** (`const key_type &__x`) `const`
- `iterator` **insert** (`const value_type &__x`)
- `iterator` **insert** (`value_type &&__x`)
- `iterator` **insert** (`const_iterator __position, const value_type &__x`)
- `iterator` **insert** (`const_iterator __position, value_type &&__x`)
- `template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`void` **insert** (`_InputIterator __first, _InputIterator __last`)
- `void` **insert** (`initializer_list< value_type > __l`)
- `iterator` **lower\_bound** (`const key_type &__x`)
- `const_iterator` **lower\_bound** (`const key_type &__x`) `const`
- `multiset &` **operator=** (`const multiset &__x`)
- `multiset &` **operator=** (`multiset &&__x`)
- `multiset &` **operator=** (`initializer_list< value_type > __l`)
- `reverse_iterator` **rbegin** () `noexcept`
- `const_reverse_iterator` **rbegin** () `const` `noexcept`
- `reverse_iterator` **rend** () `noexcept`
- `const_reverse_iterator` **rend** () `const` `noexcept`
- `void` **swap** (`multiset &__x`)
- `iterator` **upper\_bound** (`const key_type &__x`)
- `const_iterator` **upper\_bound** (`const key_type &__x`) `const`

#### 4.517.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__profile-
::multiset< _Key, _Compare, _Allocator >
```

Class `std::multiset` wrapper with performance instrumentation.

Definition at line 41 of file `profile/multiset.h`.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

#### 4.518 `std::__profile::set<_Key, _Compare, _Allocator>` Class Template Reference

Inherits `set<_Key, _Compare, _Allocator>`.

## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **set** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` , typename = `std::_RequireInputIter<_InputIterator>>`  
**set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **set** (const `set` &\_\_x)
- **set** (const `_Base` &\_\_x)
- **set** (`set` &&\_\_x) noexcept(`is_nothrow_copy_constructible<_Compare>` )
- **set** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **end** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... `_Args`>  
`std::pair`< iterator, bool > **emplace** (`_Args` &&...\_\_args)
- template<typename... `_Args`>  
iterator **emplace\_hint** (const\_iterator \_\_pos, `_Args` &&...\_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- `std::pair`< iterator, iterator > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > **equal\_range** (const `key_type` &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)

- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_x)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- [set](#) & **operator=** (const [set](#) &\_\_x)
- [set](#) & **operator=** ([set](#) &&\_\_x)
- [set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** ([set](#) &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 4.518.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>> class std::\_profile-  
::set< \_Key, \_Compare, \_Allocator >

Class std::set wrapper with performance instrumentation.

Definition at line 41 of file profile/set.h.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

## 4.519 `std::_profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Inherits `unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`.

### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- typedef \_Base::iterator **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_Base::mapped\_type **mapped\_type**
- typedef \_Base::reference **reference**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const [unordered\\_map](#) &\_\_x)
- **unordered\_map** (const [\\_Base](#) &\_\_x)
- **unordered\_map** ([unordered\\_map](#) &&\_\_x)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void **clear** () noexcept
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_it, \_Args &&... \_\_args)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
[std::pair](#)< iterator, bool > **insert** (\_Pair &&\_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_iter, \_Pair &&\_\_v)
- template<typename \_InputIter >  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- void **insert** (const value\_type \*\_\_first, const value\_type \*\_\_last)
- [unordered\\_map](#) & **operator=** (const [unordered\\_map](#) &\_\_x)
- [unordered\\_map](#) & **operator=** ([unordered\\_map](#) &&\_\_x)
- [unordered\\_map](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- mapped\_type & **operator[]** (const \_Key &\_\_k)
- mapped\_type & **operator[]** (\_Key &&\_\_k)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_map](#) &\_\_x)

## 4.519.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<_Key>> class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_map` wrapper with performance instrumentation.

Definition at line 50 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

4.520 `std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Inherits `unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`.

## Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const [unordered\\_multimap](#) &\_\_x)
- **unordered\_multimap** (const [\\_Base](#) &\_\_x)
- **unordered\_multimap** ([unordered\\_multimap](#) && \_\_x)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void **clear** () noexcept
- `template<typename... _Args>`  
 iterator **emplace** (\_Args &&... \_\_args)
- `template<typename... _Args>`  
 iterator **emplace\_hint** (const\_iterator \_\_it, \_Args &&... \_\_args)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
 iterator **insert** (\_Pair && \_\_obj)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
 iterator **insert** (const\_iterator \_\_iter, \_Pair && \_\_v)
- `template<typename _InputIter >`  
 void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- void **insert** (const value\_type \* \_\_first, const value\_type \* \_\_last)
- [unordered\\_multimap](#) & **operator=** (const [unordered\\_multimap](#) & \_\_x)
- [unordered\\_multimap](#) & **operator=** ([unordered\\_multimap](#) && \_\_x)
- [unordered\\_multimap](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_multimap](#) & \_\_x)

## 4.520.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<_Key>> class std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multimap` wrapper with performance instrumentation.

Definition at line 352 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

4.521 `std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Inherits `unordered_multiset< _Value, _Hash, _Pred, _Alloc >`.

## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_multiset** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`const unordered_multiset &__x`)
- **unordered\_multiset** (`const _Base &__x`)
- **unordered\_multiset** (`unordered_multiset &&__x`)
- **unordered\_multiset** (`initializer_list< value_type > __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `void clear ()` noexcept
- `template<typename... _Args>`  
`iterator emplace (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __it, _Args &&... __args)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const value_type &__obj)`
- `iterator insert (const_iterator __iter, const value_type &__v)`

- iterator **insert** (value\_type &&\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, value\_type &&\_\_v)
- template<typename \_InputIter >  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- void **insert** (const value\_type \* \_\_first, const value\_type \* \_\_last)
- `unordered_multiset` & **operator=** (const `unordered_multiset` & \_\_x)
- `unordered_multiset` & **operator=** (`unordered_multiset` && \_\_x)
- `unordered_multiset` & **operator=** (initializer\_list< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** (`unordered_multiset` & \_\_x)

#### 4.521.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc =
std::allocator<_Value>> class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

`Unordered_multiset` wrapper with performance instrumentation.

Definition at line 318 of file `profile/unordered_set`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 4.522 `std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >` Class Template Reference

Inherits `unordered_set< _Key, _Hash, _Pred, _Alloc >`.

### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_set** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const `unordered_set` & \_\_x)

- **unordered\_set** (const [\\_Base](#) &\_\_x)
- **unordered\_set** ([unordered\\_set](#) &&\_\_x)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void **clear** () noexcept
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > **emplace** (\_Args &&...\_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_it, \_Args &&...\_\_args)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, value\_type &&\_\_v)
- template<typename \_InputIter >  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- void **insert** (const value\_type \*\_\_first, const value\_type \*\_\_last)
- [unordered\\_set](#) & **operator=** (const [unordered\\_set](#) &\_\_x)
- [unordered\\_set](#) & **operator=** ([unordered\\_set](#) &&\_\_x)
- [unordered\\_set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_set](#) &\_\_x)

#### 4.522.1 Detailed Description

template<typename \_Key, typename \_Hash = std::hash<\_Key>, typename \_Pred = std::equal\_to<\_Key>, typename \_Alloc = std::allocator<\_Key>> class std::\_profile::unordered\_set< \_Key, \_Hash, \_Pred, \_Alloc >

Unordered\_set wrapper with performance instrumentation.

Definition at line 50 of file profile/unordered\_set.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 4.523 std::\_Base\_bitset< \_Nw > Struct Template Reference

### Public Types

- typedef unsigned long **\_WordT**

### Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
bool **\_M\_are\_all** () const noexcept
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- size\_t **\_M\_do\_count** () const noexcept
- size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept

- `void _M_do_flip () noexcept`
- `void _M_do_left_shift (size_t __shift) noexcept`
- `void _M_do_or (const \_Base\_bitset<_Nw> &__x) noexcept`
- `void _M_do_reset () noexcept`
- `void _M_do_right_shift (size_t __shift) noexcept`
- `void _M_do_set () noexcept`
- `unsigned long long _M_do_to_ullong () const`
- `unsigned long _M_do_to_ulong () const`
- `void _M_do_xor (const \_Base\_bitset<_Nw> &__x) noexcept`
- `const _WordT * _M_getdata () const noexcept`
- `_WordT & _M_getword (size_t __pos) noexcept`
- `constexpr _WordT _M_getword (size_t __pos) const noexcept`
- `_WordT & _M_hiword () noexcept`
- `constexpr _WordT _M_hiword () const noexcept`
- `bool _M_is_any () const noexcept`
- `bool _M_is_equal (const \_Base\_bitset<_Nw> &__x) const noexcept`

#### Static Public Member Functions

- `static constexpr _WordT _S_maskbit (size_t __pos) noexcept`
- `static constexpr size_t _S_whichbit (size_t __pos) noexcept`
- `static constexpr size_t _S_whichbyte (size_t __pos) noexcept`
- `static constexpr size_t _S_whichword (size_t __pos) noexcept`

#### Public Attributes

- `_WordT \_M\_w[_Nw]`

#### 4.523.1 Detailed Description

`template<size_t _Nw> struct std::_Base_bitset<_Nw>`

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.

See documentation for `bitset`.

Definition at line 71 of file `bitset`.

#### 4.523.2 Member Data Documentation

4.523.2.1 `template<size_t _Nw> _WordT std::_Base_bitset<_Nw>::_M_w[_Nw]`

0 is the least significant word.

Definition at line 76 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

4.524 `std::_Base_bitset< 0 >` Struct Template Reference

## Public Types

- typedef unsigned long **\_WordT**

## Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long) noexcept
- template<size\_t \_Nb>  
  bool **\_M\_are\_all** () const noexcept
- void **\_M\_do\_and** (const [\\_Base\\_bitset< 0 >](#) &) noexcept
- size\_t **\_M\_do\_count** () const noexcept
- size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept
- void **\_M\_do\_flip** () noexcept
- void **\_M\_do\_left\_shift** (size\_t) noexcept
- void **\_M\_do\_or** (const [\\_Base\\_bitset< 0 >](#) &) noexcept
- void **\_M\_do\_reset** () noexcept
- void **\_M\_do\_right\_shift** (size\_t) noexcept
- void **\_M\_do\_set** () noexcept
- unsigned long long **\_M\_do\_to\_ullong** () const noexcept
- unsigned long **\_M\_do\_to\_ulong** () const noexcept
- void **\_M\_do\_xor** (const [\\_Base\\_bitset< 0 >](#) &) noexcept
- **\_WordT** & **\_M\_getword** (size\_t) noexcept
- constexpr **\_WordT** **\_M\_getword** (size\_t \_\_pos) const noexcept
- constexpr **\_WordT** **\_M\_hiword** () const noexcept
- bool **\_M\_is\_any** () const noexcept
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset< 0 >](#) &) const noexcept

## Static Public Member Functions

- static constexpr **\_WordT** **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept

## 4.524.1 Detailed Description

template<>struct std::\_Base\_bitset< 0 >

Base class, specialization for no storage (zero-length bitset).

See documentation for bitset.

Definition at line 519 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

4.525 `std::_Base_bitset< 1 >` Struct Template Reference

## Public Types

- typedef unsigned long `_WordT`

## Public Member Functions

- constexpr `_Base_bitset` (unsigned long long `__val`) noexcept
- template<size\_t `Nb`>  
  bool `_M_are_all` () const noexcept
- void `_M_do_and` (const `_Base_bitset< 1 >` &`__x`) noexcept
- size\_t `_M_do_count` () const noexcept
- size\_t `_M_do_find_first` (size\_t `__not_found`) const noexcept
- size\_t `_M_do_find_next` (size\_t `__prev`, size\_t `__not_found`) const noexcept
- void `_M_do_flip` () noexcept
- void `_M_do_left_shift` (size\_t `__shift`) noexcept
- void `_M_do_or` (const `_Base_bitset< 1 >` &`__x`) noexcept
- void `_M_do_reset` () noexcept
- void `_M_do_right_shift` (size\_t `__shift`) noexcept
- void `_M_do_set` () noexcept
- unsigned long long `_M_do_to_ullong` () const noexcept
- unsigned long `_M_do_to_ulong` () const noexcept
- void `_M_do_xor` (const `_Base_bitset< 1 >` &`__x`) noexcept
- const `_WordT` \* `_M_getdata` () const noexcept
- `_WordT` & `_M_getword` (size\_t) noexcept
- constexpr `_WordT` `_M_getword` (size\_t) const noexcept
- `_WordT` & `_M_hiword` () noexcept
- constexpr `_WordT` `_M_hiword` () const noexcept
- bool `_M_is_any` () const noexcept
- bool `_M_is_equal` (const `_Base_bitset< 1 >` &`__x`) const noexcept

## Static Public Member Functions

- static constexpr `_WordT` `_S_maskbit` (size\_t `__pos`) noexcept
- static constexpr size\_t `_S_whichbit` (size\_t `__pos`) noexcept
- static constexpr size\_t `_S_whichbyte` (size\_t `__pos`) noexcept
- static constexpr size\_t `_S_whichword` (size\_t `__pos`) noexcept

## Public Attributes

- `_WordT` `_M_w`

## 4.525.1 Detailed Description

```
template<> struct std::_Base_bitset< 1 >
```

Base class, specialization for a single word.

See documentation for `bitset`.

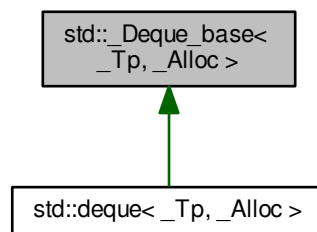
Definition at line 372 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 4.526 std::\_Deque\_base&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for `std::_Deque_base< _Tp, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Deque_iterator< _Tp, const _Tp &, const _Tp * >` **const\_iterator**
- typedef `_Deque_iterator< _Tp, _Tp &, _Tp * >` **iterator**

## Public Member Functions

- `_Deque_base` (`size_t __num_elements`)
- `_Deque_base` (`const allocator_type &__a, size_t __num_elements`)
- `_Deque_base` (`const allocator_type &__a`)
- `_Deque_base` (`_Deque_base &&__x`)
- `allocator_type` **get\_allocator** () `const noexcept`

## Protected Types

- enum { **\_S\_initial\_map\_size** }
- typedef \_Alloc::template rebind<\_Tp \* >::other **\_Map\_alloc\_type**
- typedef \_Alloc::template rebind<\_Tp >::other **\_Tp\_alloc\_type**

## Protected Member Functions

- **\_Tp\*\* \_M\_allocate\_map** (size\_t \_\_n)
- **\_Tp\* \_M\_allocate\_node** ()
- void **\_M\_create\_nodes** (\_Tp\*\* \_\_nstart, \_Tp\*\* \_\_nfinish)
- void **\_M\_deallocate\_map** (\_Tp\*\* \_\_p, size\_t \_\_n)
- void **\_M\_deallocate\_node** (\_Tp\* \_\_p)
- void **\_M\_destroy\_nodes** (\_Tp\*\* \_\_nstart, \_Tp\*\* \_\_nfinish)
- **\_Map\_alloc\_type \_M\_get\_map\_allocator** () const noexcept
- **\_Tp\_alloc\_type & \_M\_get\_Tp\_allocator** () noexcept
- const **\_Tp\_alloc\_type & \_M\_get\_Tp\_allocator** () const noexcept
- void **\_M\_initialize\_map** (size\_t)

## Protected Attributes

- **\_Deque\_impl \_M\_impl**

## 4.526.1 Detailed Description

template<typename \_Tp, typename \_Alloc>class std::\_Deque\_base<\_Tp, \_Alloc >

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 439 of file stl\_deque.h.

## 4.526.2 Member Function Documentation

4.526.2.1 template<typename \_Tp, typename \_Alloc > void std::\_Deque\_base<\_Tp, \_Alloc >::M\_initialize\_map ( size\_t \_\_num\_elements ) [protected]

Layout storage.

## Parameters

|                       |                                                        |
|-----------------------|--------------------------------------------------------|
| <b>__num_elements</b> | The count of T's for which to allocate space at first. |
|-----------------------|--------------------------------------------------------|

## Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 582 of file `stl_deque.h`.

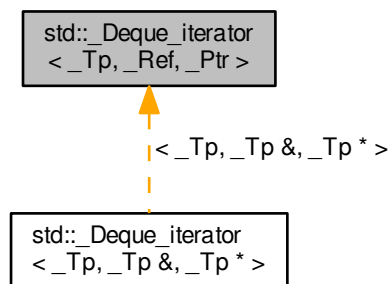
References `std::max()`.

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

4.527 `std::_Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference

Inheritance diagram for `std::_Deque_iterator<_Tp, _Ref, _Ptr>`:



## Public Types

- `typedef _Tp** Map_pointer`
- `typedef \_Deque\_iterator Self`
- `typedef \_Deque\_iterator<_Tp, const _Tp &, const _Tp*> const_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef \_Deque\_iterator<_Tp, _Tp &, _Tp*> iterator`
- `typedef std::random\_access\_iterator\_tag iterator_category`
- `typedef _Ptr pointer`
- `typedef _Ref reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `_Deque_iterator` (`_Tp *__x`, `_Map_pointer __y`)
- `_Deque_iterator` (`const iterator &__x`)
- `void _M_set_node` (`_Map_pointer __new_node`)
- reference `operator*` () `const`
- `_Self operator+` (`difference_type __n`) `const`
- `_Self & operator++` ()
- `_Self operator++` (`int`)
- `_Self & operator+=` (`difference_type __n`)
- `_Self operator-` (`difference_type __n`) `const`
- `_Self & operator--` ()
- `_Self operator--` (`int`)
- `_Self & operator-=` (`difference_type __n`)
- pointer `operator->` () `const`
- reference `operator[]` (`difference_type __n`) `const`

## Static Public Member Functions

- static `size_t _S_buffer_size` ()

## Public Attributes

- `_Tp * _M_cur`
- `_Tp * _M_first`
- `_Tp * _M_last`
- `_Map_pointer _M_node`

## 4.527.1 Detailed Description

```
template<typename _Tp, typename _Ref, typename _Ptr>struct std::Deque_iterator< _Tp, _Ref, _Ptr >
```

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 106 of file `stl_deque.h`.

## 4.527.2 Member Function Documentation

4.527.2.1 `template<typename _Tp, typename _Ref, typename _Ptr> void std::Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node`  
`( _Map_pointer __new_node ) [inline]`

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

Definition at line 234 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl\\_deque.h](#)

## 4.528 `std::_Derives_from_binary_function<_Tp>` Struct Template Reference

Inherits `std::__sfinae_types`.

### Public Types

- typedef char **\_\_one**

### Static Public Attributes

- static const bool **value**

#### 4.528.1 Detailed Description

```
template<typename _Tp>struct std::_Derives_from_binary_function<_Tp>
```

Determines if the type `_Tp` derives from `binary_function`.

Definition at line 206 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.529 `std::_Derives_from_unary_function<_Tp>` Struct Template Reference

Inherits `std::__sfinae_types`.

### Public Types

- typedef char **\_\_one**

### Static Public Attributes

- static const bool **value**

#### 4.529.1 Detailed Description

```
template<typename _Tp>struct std::_Derives_from_unary_function<_Tp>
```

Determines if the type `_Tp` derives from `unary_function`.

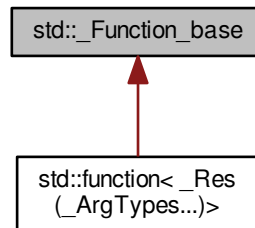
Definition at line 190 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.530 std::\_Function\_base Class Reference

Inheritance diagram for std::\_Function\_base:



### Public Types

- `typedef bool(* _Manager_type)(_Any_data &, const _Any_data &, _Manager_operation)`

### Public Member Functions

- `bool _M_empty () const`

### Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

### Static Public Attributes

- `static const std::size_t _M_max_align`
- `static const std::size_t _M_max_size`

#### 4.530.1 Detailed Description

Base class of all polymorphic function object wrappers.

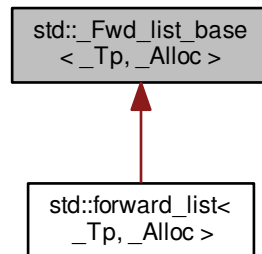
Definition at line 1869 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 4.531 std::\_Fwd\_list\_base&lt; \_Tp, \_Alloc &gt; Struct Template Reference

Inheritance diagram for std::\_Fwd\_list\_base< \_Tp, \_Alloc >:



## Public Types

- typedef `_Fwd_list_node< _Tp >` **\_Node**
- typedef `_Fwd_list_const_iterator< _Tp >` **const\_iterator**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**

## Public Member Functions

- **\_Fwd\_list\_base** (const `_Node_alloc_type` &\_\_a)
- **\_Fwd\_list\_base** (`_Fwd_list_base` &&\_\_lst, const `_Node_alloc_type` &\_\_a)
- **\_Fwd\_list\_base** (`_Fwd_list_base` &&\_\_lst)
- `_Node_alloc_type` & **\_M\_get\_Node\_allocator** () noexcept
- const `_Node_alloc_type` & **\_M\_get\_Node\_allocator** () const noexcept

## Protected Types

- typedef `__gnu_cxx::__alloc_traits< _Alloc >` **\_Alloc\_traits**
- typedef `__gnu_cxx::__alloc_traits< _Node_alloc_type >` **\_Node\_alloc\_traits**
- typedef `_Alloc_traits::template rebind< _Fwd_list_node< _Tp >>::other` **\_Node\_alloc\_type**
- typedef `_Alloc_traits::template rebind< _Tp >::other` **\_Tp\_alloc\_type**

## Protected Member Functions

- `template<typename... _Args>`  
`_Node * _M_create_node ( _Args &&...__args)`
- `_Fwd_list_node_base * _M_erase_after ( _Fwd_list_node_base * __pos)`
- `_Fwd_list_node_base * _M_erase_after ( _Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `template<typename... _Args>`  
`_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&...__args)`
- `void _M_put_node ( _Node * __p)`

## Protected Attributes

- `_Fwd_list_impl _M_impl`

## 4.531.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::_Fwd_list_base< _Tp, _Alloc >`

Base class for forward\_list.

Definition at line 275 of file forward\_list.h.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 4.532 std::\_Fwd\_list\_const\_iterator&lt; \_Tp &gt; Struct Template Reference

## Public Types

- `typedef const _Fwd_list_node< _Tp > _Node`
- `typedef`  
`_Fwd_list_const_iterator< _Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _Fwd_list_iterator< _Tp > iterator`
- `typedef std::forward_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

## Public Member Functions

- `_Fwd_list_const_iterator (const _Fwd_list_node_base * __n)`
- `_Fwd_list_const_iterator (const iterator & __iter)`
- `_Self _M_next () const`
- `bool operator!= (const _Self & __x) const`
- `reference operator* () const`
- `_Self & operator++ ()`
- `_Self operator++ (int)`
- `pointer operator-> () const`
- `bool operator== (const _Self & __x) const`

## Public Attributes

- [const \\_Fwd\\_list\\_node\\_base](#) \* **\_M\_node**

## 4.532.1 Detailed Description

```
template<typename _Tp>struct std::_Fwd_list_const_iterator< _Tp >
```

A forward\_list::const\_iterator.

All the functions are op overloads.

Definition at line 188 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 4.533 std::\_Fwd\_list\_iterator&lt; \_Tp &gt; Struct Template Reference

## Public Types

- typedef [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

## Public Member Functions

- [\\_Fwd\\_list\\_iterator](#) ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_n)
- [\\_Self \\_M\\_next](#) () const
- bool **operator!=** (const [\\_Self](#) & \_\_x) const
- reference **operator\*** () const
- [\\_Self](#) & **operator++** ()
- [\\_Self](#) **operator++** (int)
- pointer **operator->** () const
- bool **operator==** (const [\\_Self](#) & \_\_x) const

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_node**

## 4.533.1 Detailed Description

```
template<typename _Tp>struct std::_Fwd_list_iterator< _Tp >
```

A forward\_list::iterator.

All the functions are op overloads.

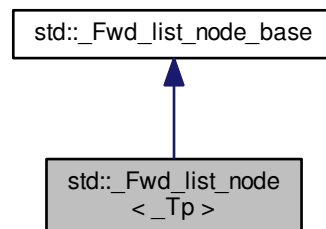
Definition at line 121 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

#### 4.534 std::\_Fwd\_list\_node< \_Tp > Struct Template Reference

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



##### Public Member Functions

- void **\_M\_reverse\_after** () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_transfer\_after** ( [\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end)
- [\\_Tp](#) \* **\_M\_valptr** () noexcept
- const [\\_Tp](#) \* **\_M\_valptr** () const noexcept

##### Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_next**
- [aligned\\_storage](#)< sizeof([\\_Tp](#)),  
[alignment\\_of](#)< [\\_Tp](#) >::value >  
::type **\_M\_storage**

##### 4.534.1 Detailed Description

```
template<typename _Tp>struct std::_Fwd_list_node< _Tp >
```

A helper node class for forward\_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

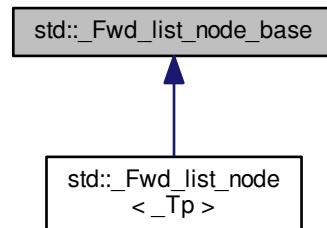
Definition at line 94 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 4.535 std::\_Fwd\_list\_node\_base Struct Reference

Inheritance diagram for std::\_Fwd\_list\_node\_base:



## Public Member Functions

- void **\_M\_reverse\_after** () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_transfer\_after** ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end)

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_next**

## 4.535.1 Detailed Description

A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

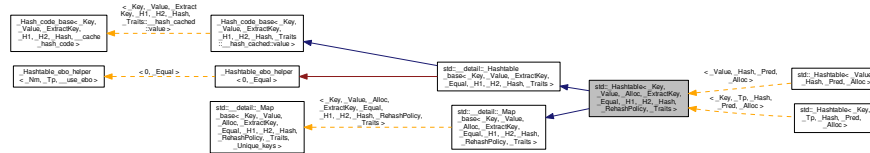
Definition at line 49 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 4.536 `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Class Template Reference

Inheritance diagram for `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- using **const\_iterator** = typename `__hashtable_base::const_iterator`
- using **const\_local\_iterator** = typename `__hashtable_base::const_local_iterator`
- typedef `_Alloc::const_pointer` **const\_pointer**
- typedef `_Alloc::const_reference` **const\_reference**
- using **difference\_type** = typename `__hashtable_base::difference_type`
- using **iterator** = typename `__hashtable_base::iterator`
- typedef `_Equal` **key\_equal**
- typedef `_Key` **key\_type**
- using **local\_iterator** = typename `__hashtable_base::local_iterator`
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- using **size\_type** = typename `__hashtable_base::size_type`
- typedef `_Value` **value\_type**

### Public Member Functions

- **\_Hashtable** (`size_type` \_\_bucket\_hint, const `_H1` &, const `_H2` &, const `_Hash` &, const `_Equal` &, const `_ExtractKey` &, const `allocator_type` &)
- template<typename `_InputIterator` >  
**\_Hashtable** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `size_type` \_\_bucket\_hint, const `_H1` &, const `_H2` &, const `_Hash` &, const `_Equal` &, const `_ExtractKey` &, const `allocator_type` &)
- **\_Hashtable** (const `_Hashtable` &)
- **\_Hashtable** (`_Hashtable` &&)
- **\_Hashtable** (`size_type` \_\_n=10, const `_H1` & \_\_hf=\_H1(), const `key_equal` & \_\_eq=key\_equal(), const `allocator_type` & \_\_a=allocator\_type())
- template<typename `_InputIterator` >  
**\_Hashtable** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, `size_type` \_\_n=0, const `_H1` & \_\_hf=\_H1(), const `key_equal` & \_\_eq=key\_equal(), const `allocator_type` & \_\_a=allocator\_type())
- **\_Hashtable** (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, const `_H1` & \_\_hf=\_H1(), const `key_equal` & \_\_eq=key\_equal(), const `allocator_type` & \_\_a=allocator\_type())
- const `_RehashPolicy` & **\_\_rehash\_policy** () const
- void **\_\_rehash\_policy** (const `_RehashPolicy` &)
- iterator **begin** () noexcept

- `const_iterator` **begin** () const noexcept
- `local_iterator` **begin** (size\_type \_\_n)
- `const_local_iterator` **begin** (size\_type \_\_n) const
- size\_type **bucket** (const key\_type &\_\_k) const
- size\_type **bucket\_count** () const noexcept
- size\_type **bucket\_size** (size\_type \_\_n) const
- `const_iterator` **cbegin** () const noexcept
- `const_local_iterator` **cbegin** (size\_type \_\_n) const
- `const_iterator` **cend** () const noexcept
- `const_local_iterator` **cend** (size\_type \_\_n) const
- void **clear** () noexcept
- size\_type **count** (const key\_type &\_\_k) const
- template<typename... \_Args>  
   \_\_ireturn\_type **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
   iterator **emplace\_hint** (const\_iterator, \_Args &&... \_\_args)
- bool **empty** () const noexcept
- iterator **end** () noexcept
- `const_iterator` **end** () const noexcept
- `local_iterator` **end** (size\_type \_\_n)
- `const_local_iterator` **end** (size\_type \_\_n) const
- `std::pair`< iterator, iterator > **equal\_range** (const key\_type &\_\_k)
- `std::pair`< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_k) const
- iterator **erase** (const\_iterator)
- iterator **erase** (iterator \_\_it)
- size\_type **erase** (const key\_type &\_\_k)
- iterator **erase** (const\_iterator, const\_iterator)
- iterator **find** (const key\_type &\_\_k)
- `const_iterator` **find** (const key\_type &\_\_k) const
- allocator\_type **get\_allocator** () const noexcept
- key\_equal **key\_eq** () const
- float **load\_factor** () const noexcept
- size\_type **max\_bucket\_count** () const noexcept
- size\_type **max\_size** () const noexcept
- `_Hashtable` & **operator=** (const `_Hashtable` &\_\_ht)
- `_Hashtable` & **operator=** (`_Hashtable` &&\_\_ht)
- `_Hashtable` & **operator=** (initializer\_list< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- size\_type **size** () const noexcept
- void **swap** (`_Hashtable` &)

#### Protected Member Functions

- size\_type **\_M\_bucket\_index** (\_\_node\_type \*\_\_n) const
- size\_type **\_M\_bucket\_index** (const key\_type &\_\_k, \_\_hash\_code \_\_c) const
- template<typename... \_Args>  
   `std::pair`< iterator, bool > **\_M\_emplace** (std::true\_type, \_Args &&... \_\_args)
- template<typename... \_Args>  
   iterator **\_M\_emplace** (std::false\_type, \_Args &&... \_\_args)
- const \_Equal & **\_M\_eq** () const

- `_Equal & _M_eq ()`
- `bool _M_equals (const _Key & __k, __hash_code __c, __node_type * __n) const`
- `size_type _M_erase (std::true_type, const key_type &)`
- `size_type _M_erase (std::false_type, const key_type &)`
- `iterator _M_erase (size_type __bkt, __node_base * __prev_n, __node_type * __n)`
- `__node_base * _M_find_before_node (size_type, const key_type &, __hash_code) const`
- `__node_type * _M_find_node (size_type __bkt, const key_type & __key, __hash_code __c) const`
- `__node_base * _M_get_previous_node (size_type __bkt, __node_base * __n)`
- `template<typename _Arg >`  
`std::pair< iterator, bool > _M_insert (_Arg &&, std::true_type)`
- `template<typename _Arg >`  
`iterator _M_insert (_Arg &&, std::false_type)`
- `void _M_insert_bucket_begin (size_type, __node_type *)`
- `iterator _M_insert_multi_node (__hash_code __code, __node_type * __n)`
- `iterator _M_insert_unique_node (size_type __bkt, __hash_code __code, __node_type * __n)`
- `void _M_remove_bucket_begin (size_type __bkt, __node_type * __next_n, size_type __next_bkt)`
- `void _M_swap (_Hashtable_base & __x)`

#### Friends

- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Constant_iteratorsa, bool _Unique_keysa>`  
`struct __detail:: Insert`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa >`  
`struct __detail:: Insert_base`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Unique_keysa>`  
`struct __detail:: Map_base`

#### 4.536.1 Detailed Description

`template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits> class std::Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Primary class template `_Hashtable`.

#### Template Parameters

|                          |                                                                                                                                                                                                                                                                               |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Value</code>      | CopyConstructible type.                                                                                                                                                                                                                                                       |
| <code>_Key</code>        | CopyConstructible type.                                                                                                                                                                                                                                                       |
| <code>_Alloc</code>      | An allocator type ([lib.allocator.requirements]) whose <code>_Alloc::value_type</code> is <code>_Value</code> . As a conforming extension, we allow for <code>_Alloc::value_type != _Value</code> .                                                                           |
| <code>_ExtractKey</code> | Function object that takes an object of type <code>_Value</code> and returns a value of type <code>_Key</code> .                                                                                                                                                              |
| <code>_Equal</code>      | Function object that takes two objects of type <code>k</code> and returns a bool-like value that is true if the two objects are considered equal.                                                                                                                             |
| <code>_H1</code>         | The hash function. A unary function object with argument type <code>_Key</code> and result type <code>size_t</code> . Return values should be distributed over the entire range <code>[0, numeric_limits&lt;size_t&gt;::max())</code> .                                       |
| <code>_H2</code>         | The range-hashing function (in the terminology of Tavori and Dreizin). A binary function object whose argument types and result type are all <code>size_t</code> . Given arguments <code>r</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> . |

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Hash</code>         | The ranged hash function (Tavori and Dreizin). A binary function whose argument types are <code>_Key</code> and <code>size_t</code> and whose result type is <code>size_t</code> . Given arguments <code>k</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> . Default: <code>hash(k, N) = h2(h1(k), N)</code> . If <code>_Hash</code> is anything other than the default, <code>_H1</code> and <code>_H2</code> are ignored.                                                                                                                                                                                                 |
| <code>_RehashPolicy</code> | Policy class with three members, all of which govern the bucket count. <code>_M_next_bkt(n)</code> returns a bucket count no smaller than <code>n</code> . <code>_M_bkt_for_elements(n)</code> returns a bucket count appropriate for an element count of <code>n</code> . <code>_M_need_rehash(n_bkt, n_elt, n_ins)</code> determines whether, if the current bucket count is <code>n_bkt</code> and the current element count is <code>n_elt</code> , we need to increase the bucket count. If so, returns <code>make_pair(true, n)</code> , where <code>n</code> is the new bucket count. If not, returns <code>make_pair(false, &lt;anything&gt;)</code> |
| <code>_Traits</code>       | Compile-time class with three boolean <code>std::integral_constant</code> members: <code>__cache_hash_code</code> , <code>__constant_iterators</code> , <code>__unique_keys</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Each `_Hashtable` data structure has:

- `_Bucket[] _M_buckets`
- `_Hash_node_base _M_bbegin`
- `size_type _M_bucket_count`
- `size_type _M_element_count`

with `_Bucket` being `_Hash_node*` and `_Hash_node` containing:

- `_Hash_node* _M_next`
- `Tp _M_value`
- `size_t _M_hash_code` if `cache_hash_code` is true

In terms of Standard containers the hashtable is like the aggregation of:

- `std::forward_list<_Node>` containing the elements
- `std::vector<std::forward_list<_Node>::iterator>` representing the buckets

The non-empty buckets contain the node before the first node in the bucket. This design makes it possible to implement something like a `std::forward_list::insert_after` on container insertion and `std::forward_list::erase_after` on container erase calls. `_M_before_begin` is equivalent to `std::forward_list::before_begin`. Empty buckets contain `nullptr`. Note that one of the non-empty buckets contains `&_M_before_begin` which is not a dereferenceable node so the node pointer in a bucket shall never be dereferenced, only its next node can be.

Walking through a bucket's nodes requires a check on the hash code to see if each node is still in the bucket. Such a design assumes a quite efficient hash functor and is one of the reasons it is highly advisable to set `__cache_hash_code` to true.

The container iterators are simply built from nodes. This way incrementing the iterator is perfectly efficient independent of how many empty buckets there are in the container.

On insert we compute the element's hash code and use it to find the bucket index. If the element must be inserted in an empty bucket we add it at the beginning of the singly linked list and make the bucket point to `_M_before_begin`. The bucket that used to point to `_M_before_begin`, if any, is updated to point to its new before begin node.

On erase, the simple iterator design requires using the hash functor to get the index of the bucket to update. For this reason, when `__cache_hash_code` is set to false the hash functor must not throw and this is enforced by a static assertion.

Functionality is implemented by decomposition into base classes, where the derived \_Hashtable class is used in \_Map\_base, \_Insert, \_Rehash\_base, and \_Equality base classes to access the "this" pointer. \_Hashtable\_base is used in the base classes as a non-recursive, fully-completed-type so that detailed nested type information, such as iterator type and node type, can be used. This is similar to the "Curiously Recurring Template Pattern" (CRTP) technique, but uses a reconstructed, not explicitly passed, template pattern.

Base class templates are:

- \_\_detail::\_Hashtable\_base
- \_\_detail::\_Map\_base
- \_\_detail::\_Insert
- \_\_detail::\_Rehash\_base
- \_\_detail::\_Equality

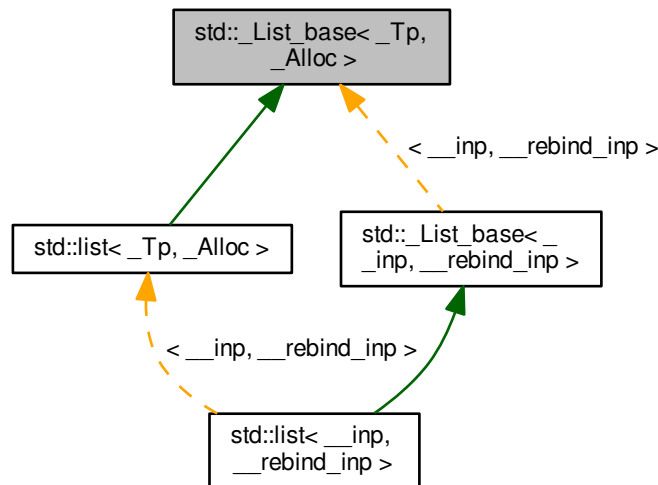
Definition at line 174 of file bits/hashtable.h.

The documentation for this class was generated from the following file:

- [bits/hashtable.h](#)

#### 4.537 std::\_List\_base< \_Tp, \_Alloc > Class Template Reference

Inheritance diagram for std::\_List\_base< \_Tp, \_Alloc >:



#### Public Types

- typedef \_Alloc **allocator\_type**

## Public Member Functions

- `_List_base` (`const _Node_alloc_type &__a`)
- `_List_base` (`_List_base &&__x`)
- `void _M_clear ()`
- `_Node_alloc_type &_M_get_Node_allocator () noexcept`
- `const _Node_alloc_type &_M_get_Node_allocator () const noexcept`
- `_Tp_alloc_type _M_get_Tp_allocator () const noexcept`
- `void _M_init ()`
- `allocator_type get_allocator () const noexcept`

## Protected Types

- `typedef _Alloc::template  
rebind<_List_node<_Tp>  
>::other _Node_alloc_type`
- `typedef _Alloc::template  
rebind<_Tp>::other _Tp_alloc_type`

## Protected Member Functions

- `_List_node<_Tp> * _M_get_node ()`
- `void _M_put_node (_List_node<_Tp> *__p)`

## Protected Attributes

- `_List_impl _M_impl`

## 4.537.1 Detailed Description

`template<typename _Tp, typename _Alloc> class std::_List_base<_Tp, _Alloc>`

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

Definition at line 289 of file `stl_list.h`.

The documentation for this class was generated from the following files:

- `stl_list.h`
- `list.tcc`

4.538 `std::_List_const_iterator<_Tp>` Struct Template Reference

## Public Types

- `typedef const _List_node<_Tp> _Node`
- `typedef _List_const_iterator<_Tp> _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _List_iterator<_Tp> iterator`
- `typedef  
std::bidirectional_iterator_tag iterator_category`

- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **List\_const\_iterator** (const \_\_detail::List\_node\_base \* \_\_x)
- **List\_const\_iterator** (const iterator & \_\_x)
- bool **operator!=** (const \_Self & \_\_x) const
- reference **operator\*** () const
- \_Self & **operator++** ()
- \_Self **operator++** (int)
- \_Self & **operator--** ()
- \_Self **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const \_Self & \_\_x) const

#### Public Attributes

- const \_\_detail::List\_node\_base \* **M\_node**

#### 4.538.1 Detailed Description

template<typename \_Tp>struct std::List\_const\_iterator< \_Tp >

A list::const\_iterator.

All the functions are op overloads.

Definition at line 200 of file stl\_list.h.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 4.539 std::List\_iterator< \_Tp > Struct Template Reference

##### Public Types

- typedef [List\\_node](#)< \_Tp > **\_Node**
- typedef [List\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

## Public Member Functions

- [\\_List\\_iterator](#) ([\\_\\_detail::\\_List\\_node\\_base](#) \*\_\_x)
- bool **operator!=** (const [\\_Self](#) &\_\_x) const
- reference **operator\*** () const
- [\\_Self](#) & **operator++** ()
- [\\_Self](#) **operator++** (int)
- [\\_Self](#) & **operator--** ()
- [\\_Self](#) **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const [\\_Self](#) &\_\_x) const

## Public Attributes

- [\\_\\_detail::\\_List\\_node\\_base](#) \* **\_M\_node**

## 4.539.1 Detailed Description

```
template<typename _Tp>struct std::_List_iterator<_Tp>
```

A list::iterator.

All the functions are op overloads.

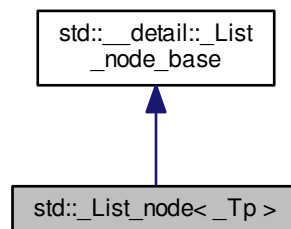
Definition at line 125 of file stl\_list.h.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 4.540 std::\_List\_node&lt;\_Tp&gt; Struct Template Reference

Inheritance diagram for std::\_List\_node<\_Tp>:



## Public Member Functions

- `template<typename... _Args>`  
`_List_node` (`_Args &&... __args`)
- `void _M_hook` (`_List_node_base *const __position`) `noexcept`
- `void _M_reverse` () `noexcept`
- `void _M_transfer` (`_List_node_base *const __first`, `_List_node_base *const __last`) `noexcept`
- `void _M_unhook` () `noexcept`

## Static Public Member Functions

- `static void swap` (`_List_node_base &__x`, `_List_node_base &__y`) `noexcept`

## Public Attributes

- `_Tp _M_data`
- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`

## 4.540.1 Detailed Description

`template<typename _Tp> struct std::_List_node< _Tp >`

An actual node in the list.

Definition at line 106 of file `stl_list.h`.

## 4.540.2 Member Data Documentation

4.540.2.1 `template<typename _Tp> _Tp std::_List_node< _Tp >::_M_data`

< User's data.

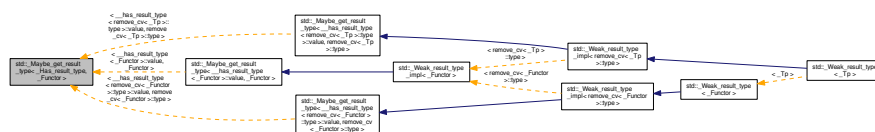
Definition at line 109 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 4.541 std::\_Maybe\_get\_result\_type&lt;\_Has\_result\_type, \_Functor &gt; Struct Template Reference

Inheritance diagram for `std::_Maybe_get_result_type<_Has_result_type, _Functor >`:



## 4.541.1 Detailed Description

```
template<bool _Has_result_type, typename _Functor>struct std::Maybe_get_result_type< _Has_result_type, _Functor >
```

If we have found a `result_type`, extract it.

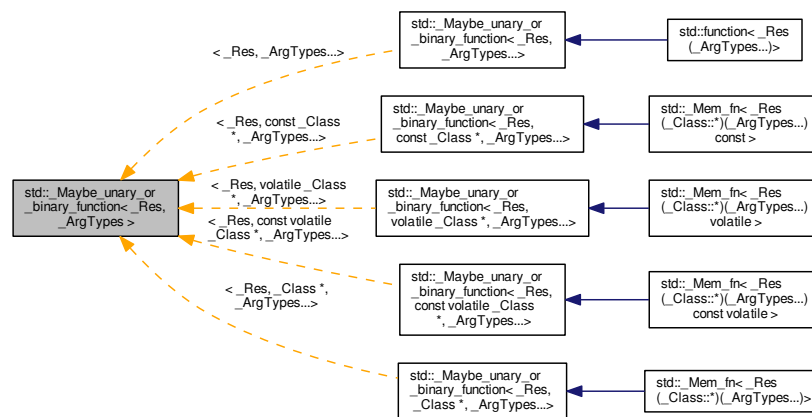
Definition at line 74 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.542 std::Maybe\_unary\_or\_binary\_function&lt;\_Res, \_ArgTypes&gt; Struct Template Reference

Inheritance diagram for `std::Maybe_unary_or_binary_function<_Res, _ArgTypes>`:



## 4.542.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>struct std::Maybe_unary_or_binary_function< _Res, _ArgTypes >
```

Derives from `unary_function` or `binary_function`, or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

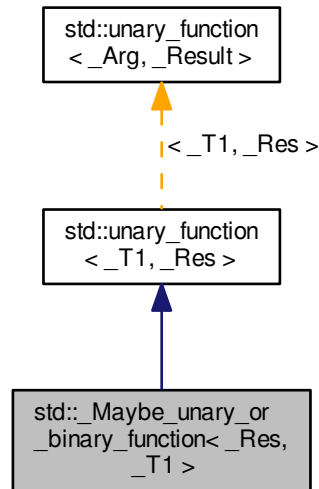
Definition at line 527 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.543 std::\_Maybe\_unary\_or\_binary\_function&lt; \_Res, \_T1 &gt; Struct Template Reference

Inheritance diagram for std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1 >:



## Public Types

- typedef `_T1` [argument\\_type](#)
- typedef `_Res` [result\\_type](#)

## 4.543.1 Detailed Description

```
template<typename _Res, typename _T1>struct std::_Maybe_unary_or_binary_function< _Res, _T1 >
```

Derives from `unary_function`, as appropriate.

Definition at line 531 of file `functional`.

## 4.543.2 Member Typedef Documentation

4.543.2.1 `typedef _T1 std::unary_function<_T1, _Res>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.543.2.2 `typedef _Res std::unary_function<_T1, _Res>::result_type` [inherited]

`result_type` is the return type

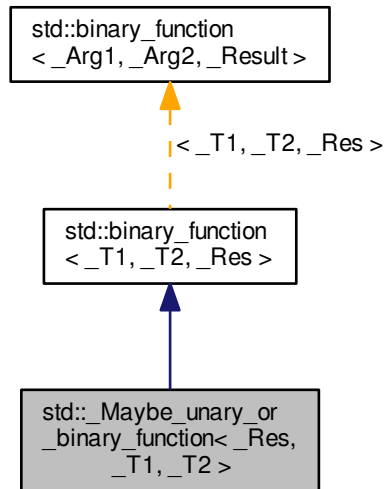
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.544 std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 > Struct Template Reference

Inheritance diagram for std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 >:



##### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `_Res` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

##### 4.544.1 Detailed Description

```
template<typename _Res, typename _T1, typename _T2>struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >
```

Derives from `binary_function`, as appropriate.

Definition at line 536 of file `functional`.

##### 4.544.2 Member Typedef Documentation

4.544.2.1 typedef `_T1` `std::binary_function<_T1, _T2, _Res>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.544.2.2 `typedef _Res std::binary_function<_T1, _T2, _Res>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.544.2.3 `typedef _T2 std::binary_function<_T1, _T2, _Res>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.545 `std::_Maybe_wrap_member_pointer<_Tp>` Struct Template Reference

### Public Types

- `typedef _Tp type`

### Static Public Member Functions

- `static const _Tp & __do_wrap (const _Tp &__x)`
- `static _Tp && __do_wrap (_Tp &&__x)`

### 4.545.1 Detailed Description

`template<typename _Tp>struct std::_Maybe_wrap_member_pointer<_Tp>`

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. The primary template handles the non-member-pointer case.

Definition at line 1223 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.546 `std::_Maybe_wrap_member_pointer<_Tp _Class::*>` Struct Template Reference

### Public Types

- `typedef _Mem_fn<_Tp _Class::*> type`

### Static Public Member Functions

- `static type __do_wrap (_Tp _Class::*__pm)`

## 4.546.1 Detailed Description

```
template<typename _Tp, typename _Class>struct std::_Maybe_wrap_member_pointer<_Tp _Class::* >
```

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. This partial specialization handles the member pointer case.

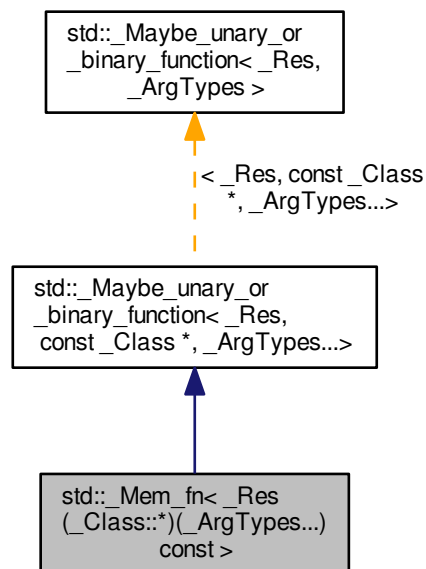
Definition at line 1242 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.547 std::\_Mem\_fn&lt;\_Res(\_Class::\*)(\_ArgTypes...) const &gt; Class Template Reference

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const >`:



## Public Types

- typedef `_Res` **result\_type**

## Public Member Functions

- **\_Mem\_fn** (`_Functor __pmf`)
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>`  
`_Res operator() (const _Class &__object, _Args &&... __args) const`

- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (const _Class &&__object, _Args &&...__args) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (const _Class *__object, _Args &&...__args) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>  
_Res operator() (_Tp &&__object, _Args &&...__args) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>  
_Res operator() (reference_wrapper<_Tp> __ref, _Args &&...__args) const`

#### 4.547.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes>class std::Mem_fn<_Res(_Class::*)(_ArgTypes...) const >`

Implementation of `mem_fn` for const member function pointers.

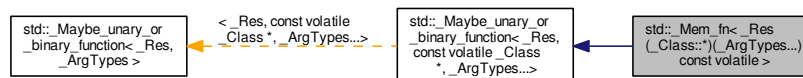
Definition at line 625 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

#### 4.548 `std::Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >` Class Template Reference

Inheritance diagram for `std::Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`:



#### Public Types

- `typedef _Res result_type`

#### Public Member Functions

- `_Mem_fn (_Functor __pmf)`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (const volatile _Class &&__object, _Args &&...__args) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (const volatile _Class &&__object, _Args &&...__args) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (const volatile _Class *__object, _Args &&...__args) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (const volatile _Class *__object, _Args &&...__args) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>  
_Res operator() (_Tp &&__object, _Args &&...__args) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>  
_Res operator() (reference_wrapper<_Tp> __ref, _Args &&...__args) const`

## 4.548.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes>class std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile
>
```

Implementation of `mem_fn` for const volatile member function pointers.

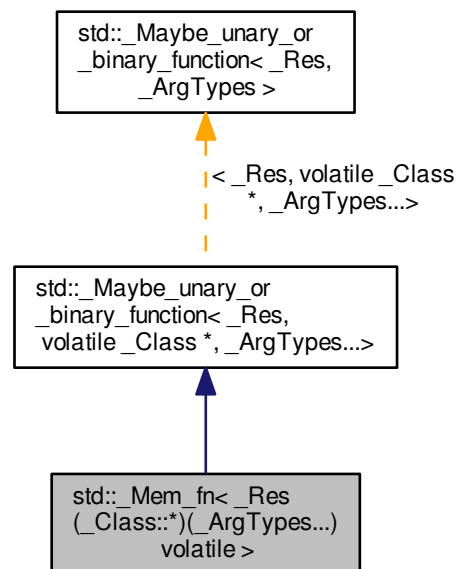
Definition at line 784 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 4.549 std::\_Mem\_fn&lt;\_Res(\_Class::\*)(\_ArgTypes...) volatile &gt; Class Template Reference

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`:



## Public Types

- `typedef _Res result_type`

## Public Member Functions

- `_Mem_fn(_Functor __pmf)`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>> _Res operator() (volatile _Class &__object, _Args &&...__args) const`

- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (volatile _Class &&__object, _Args &&...__args) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() (volatile _Class *__object, _Args &&...__args) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>  
_Res operator() (_Tp &&__object, _Args &&...__args) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>  
_Res operator() (reference_wrapper<_Tp> __ref, _Args &&...__args) const`

#### 4.549.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes>class std::Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`

Implementation of `mem_fn` for volatile member function pointers.

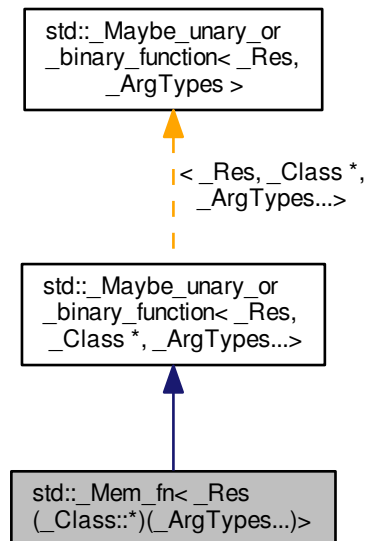
Definition at line 704 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

#### 4.550 std::Mem\_fn<\_Res(\_Class::\*)(\_ArgTypes...)> Class Template Reference

Inheritance diagram for `std::Mem_fn<_Res(_Class::*)(_ArgTypes...)>`:



#### Public Types

- `typedef _Res result_type`

## Public Member Functions

- `_Mem_fn ( Functor __pmf )`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() ( _Class &__object, _Args &&...__args ) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() ( _Class &&__object, _Args &&...__args ) const`
- `template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>  
_Res operator() ( _Class *__object, _Args &&...__args ) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>  
_Res operator() ( _Tp &&__object, _Args &&...__args ) const`
- `template<typename _Tp, typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>  
_Res operator() ( reference\_wrapper<_Tp> __ref, _Args &&...__args ) const`

## 4.550.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes>class std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>
```

Implementation of `mem_fn` for member function pointers.

Definition at line 541 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

4.551 `std::_Mu<_Arg, false, false >` Class Template Reference

## Public Member Functions

- `template<typename _CVarArg, typename _Tuple >  
_CVarArg && operator() ( _CVarArg &&__arg, _Tuple & ) const volatile`

## 4.551.1 Detailed Description

```
template<typename _Arg>class std::_Mu<_Arg, false, false >
```

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are the same as the cv-qualifiers on the `_Mu` object. [TR1 3.6.3/5 bullet 4]

Definition at line 1199 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

4.552 `std::_Mu<_Arg, false, true >` Class Template Reference

## Public Member Functions

- `template<typename _Tuple >  
result< \_Mu(_Arg, _Tuple)>::type operator() (const volatile _Arg &, _Tuple &__tuple) const volatile`

## 4.552.1 Detailed Description

```
template<typename _Arg>class std::_Mu< _Arg, false, true >
```

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. [TR1 3.6.3/5 bullet 3]

Definition at line 1165 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

4.553 `std::_Mu<_Arg, true, false >` Class Template Reference

## Public Member Functions

- `template<typename _CVArg, typename... _Args>  
auto operator() (_CVArg &__arg, tuple< _Args...> &__tuple) const volatile-> decltype(__arg(declval< _Args>()...))`

## 4.553.1 Detailed Description

```
template<typename _Arg>class std::_Mu< _Arg, true, false >
```

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). [TR1 3.6.3/5 bullet 2]

Definition at line 1131 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

4.554 `std::_Mu<reference_wrapper<_Tp>, false, false >` Class Template Reference

## Public Types

- `typedef _Tp & result_type`

## Public Member Functions

- `template<typename _CVRef, typename _Tuple >  
result_type operator() (_CVRef &__arg, _Tuple &) const volatile`

## 4.554.1 Detailed Description

```
template<typename _Tp>class std::_Mu< reference_wrapper<_Tp>, false, false >
```

If the argument is `reference_wrapper<_Tp>`, returns the underlying reference. [TR1 3.6.3/5 bullet 1]

Definition at line 1110 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 4.555 std::\_Placeholder<\_Num> Struct Template Reference

### 4.555.1 Detailed Description

template<int \_Num>struct std::\_Placeholder<\_Num>

The type of placeholder objects defined by libstdc++.

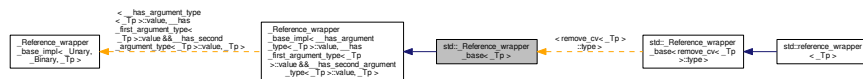
Definition at line 989 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.556 std::\_Reference\_wrapper\_base<\_Tp> Struct Template Reference

Inheritance diagram for std::\_Reference\_wrapper\_base<\_Tp>:



### 4.556.1 Detailed Description

template<typename \_Tp>struct std::\_Reference\_wrapper\_base<\_Tp>

Derives from unary\_function or binary\_function when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both unary\_function and binary\_function.

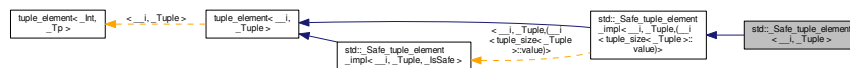
Definition at line 315 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.557 std::\_Safe\_tuple\_element<\_\_i,\_Tuple> Struct Template Reference

Inheritance diagram for std::\_Safe\_tuple\_element<\_\_i,\_Tuple>:



## 4.557.1 Detailed Description

```
template<std::size_t __i, typename _Tuple>struct std::_Safe_tuple_element< __i, _Tuple >
```

Like tuple\_element, but returns \_No\_tuple\_element when tuple\_element would return an error.

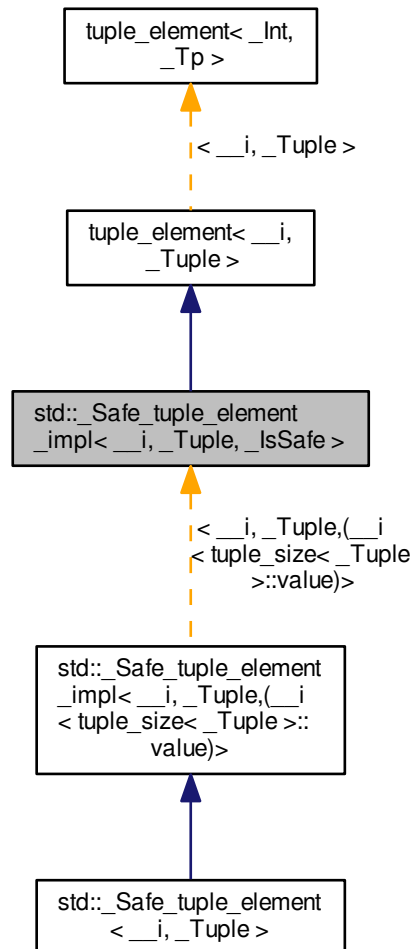
Definition at line 1084 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.558 std::\_Safe\_tuple\_element\_impl&lt; \_\_i, \_Tuple, \_IsSafe &gt; Struct Template Reference

Inheritance diagram for std::\_Safe\_tuple\_element\_impl< \_\_i, \_Tuple, \_IsSafe >:



## 4.558.1 Detailed Description

```
template<std::size_t __i, typename _Tuple, bool _IsSafe>struct std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >
```

Implementation helper for `_Safe_tuple_element`. This primary template handles the case where it is safe to use `tuple_element`.

Definition at line 1065 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.559 `std::_Safe_tuple_element_impl< __i, _Tuple, false >` Struct Template Reference

## Public Types

- typedef `_No_tuple_element` **type**

## 4.559.1 Detailed Description

```
template<std::size_t __i, typename _Tuple>struct std::_Safe_tuple_element_impl< __i, _Tuple, false >
```

Implementation helper for `_Safe_tuple_element`. This partial specialization handles the case where it is not safe to use `tuple_element`. We just return `_No_tuple_element`.

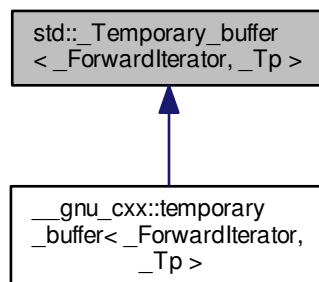
Definition at line 1074 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.560 `std::_Temporary_buffer< _ForwardIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::_Temporary_buffer< _ForwardIterator, _Tp >`:



## Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- [\\_Temporary\\_buffer](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

## Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

## 4.560.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>class std::_Temporary_buffer< _ForwardIterator, _Tp >
```

This class is used in two places: stl\_algo.h and ext/memory, where it is wrapped as the temporary\_buffer class. See temporary\_buffer docs for more notes.

Definition at line 122 of file stl\_tempbuf.h.

## 4.560.2 Constructor &amp; Destructor Documentation

4.560.2.1 `template<typename _ForwardIterator, typename _Tp> std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer ( _ForwardIterator __first, _ForwardIterator __last )`

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file stl\_tempbuf.h.

References `std::pair< _T1, _T2 >::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair< _T1, _T2 >::second`.

## 4.560.3 Member Function Documentation

4.560.3.1 `template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( ) [inline]`

As per Table mumble.

Definition at line 151 of file stl\_tempbuf.h.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

4.560.3.2 `template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer<_ForwardIterator, _Tp>::end( ) [inline]`

As per Table mumble.

Definition at line 156 of file `stl_tempbuf.h`.

4.560.3.3 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size( ) const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 146 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

4.560.3.4 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer<_ForwardIterator, _Tp>::size( ) const [inline]`

As per Table mumble.

Definition at line 141 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

## 4.561 `std::_Tuple_impl<_Idx>` Struct Template Reference

### Public Member Functions

- `template<typename _Alloc> _Tuple_impl(allocator_arg_t, const _Alloc &)`
- `template<typename _Alloc> _Tuple_impl(allocator_arg_t, const _Alloc &, const _Tuple_impl &)`
- `template<typename _Alloc> _Tuple_impl(allocator_arg_t, const _Alloc &, _Tuple_impl &&)`

### Protected Member Functions

- `void _M_swap(_Tuple_impl &) noexcept`

### Friends

- `template<std::size_t _Idx, typename...> class _Tuple_impl`

### 4.561.1 Detailed Description

`template<std::size_t _Idx> struct std::_Tuple_impl<_Idx>`

Zero-element tuple implementation. This is the basis case for the inheritance recursion.

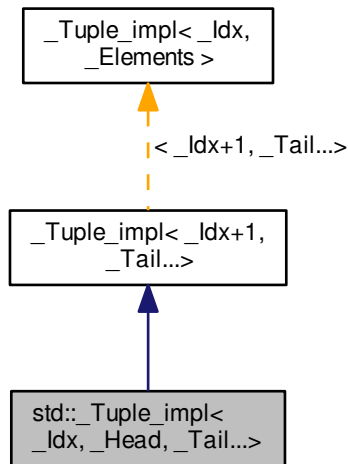
Definition at line 191 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

#### 4.562 std::\_Tuple\_impl< \_Idx, \_Head, \_Tail...> Struct Template Reference

Inheritance diagram for std::\_Tuple\_impl< \_Idx, \_Head, \_Tail...>:



#### Public Types

- `typedef _Head_base< _Idx, _Head, __empty_not_final < _Head >::value > _Base`
- `typedef \_Tuple\_impl< _Idx+1, _Tail...> _Inherited`

#### Public Member Functions

- `constexpr _Tuple_impl (const _Head &__head, const _Tail &...__tail)`
- `template<typename _UHead, typename... _UTail, typename = typename enable_if<sizeof...(_Tail) == sizeof...(_UTail)>::type> constexpr _Tuple_impl (_UHead &&__head, _UTail &&...__tail)`
- `constexpr _Tuple_impl (const \_Tuple\_impl &)=default`
- `constexpr _Tuple_impl (\_Tuple\_impl &&__in) noexcept(__and< is\_nothrow\_move\_constructible< _Head >`
- `template<typename... _UElements> constexpr _Tuple_impl (const \_Tuple\_impl< _Idx, _UElements...> &__in)`
- `template<typename _UHead, typename... _UTails> constexpr _Tuple_impl (\_Tuple\_impl< _Idx, _UHead, _UTails...> &&__in)`

- `template<typename _Alloc >`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc >`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, const _Head &__head, const _Tail &...__tail)`
- `template<typename _Alloc, typename _UHead, typename... _UTail, typename = typename enable_if<sizeof...(_Tail) == sizeof...(_UTail)>::type>`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, _UHead &&__head, _UTail &&...__tail)`
- `template<typename _Alloc >`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, const _Tuple_impl &__in)`
- `template<typename _Alloc >`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, _Tuple_impl &&__in)`
- `template<typename _Alloc, typename... _UElements>`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, const _Tuple_impl<_Idx, _UElements...> &__in)`
- `template<typename _Alloc, typename _UHead, typename... _UTails>`  
`_Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, _Tuple_impl<_Idx, _UHead, _UTails...> &&__in)`
- `constexpr`  
`is_nothrow_move_constructible`  
`<_Inherited> > _Base (std::forward<_Head>(_M_head(__in)))`
- `_Tuple_impl & operator= (const _Tuple_impl &__in)`
- `_Tuple_impl & operator= (_Tuple_impl &&__in) noexcept(__and_< is_nothrow_move_assignable<_Head>`
- `template<typename... _UElements>`  
`_Tuple_impl & operator= (const _Tuple_impl<_Idx, _UElements...> &__in)`
- `template<typename _UHead, typename... _UTails>`  
`_Tuple_impl & operator= (_Tuple_impl<_Idx, _UHead, _UTails...> &&__in)`

#### Static Public Member Functions

- `static constexpr _Head & _M_head (_Tuple_impl &__t) noexcept`
- `static constexpr const _Head & _M_head (const _Tuple_impl &__t) noexcept`
- `static constexpr _Inherited & _M_tail (_Tuple_impl &__t) noexcept`
- `static constexpr const _Inherited & _M_tail (const _Tuple_impl &__t) noexcept`

#### Public Attributes

- `_M_tail this`
- `return * this`

#### Protected Member Functions

- `void _M_swap (_Tuple_impl &__in) noexcept(noexcept(swap(std`

#### Friends

- `template<std::size_t, typename... >`  
`class _Tuple_impl`

## 4.562.1 Detailed Description

```
template<std::size_t _Idx, typename _Head, typename... _Tail>struct std::_Tuple_impl<_Idx, _Head, _Tail...>
```

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

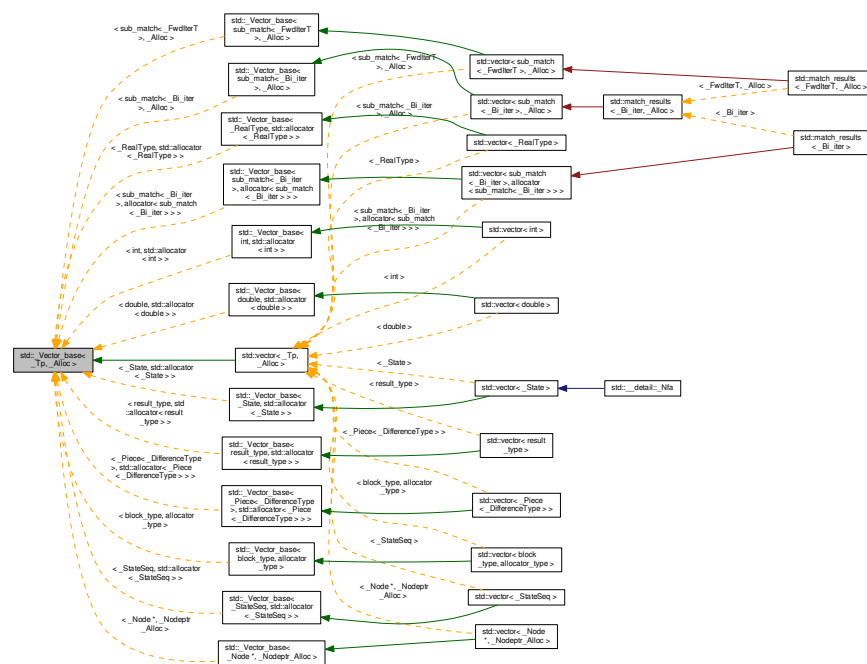
Definition at line 229 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.563 std::\_Vector\_base&lt;\_Tp, \_Alloc&gt; Struct Template Reference

Inheritance diagram for `std::_Vector_base<_Tp, _Alloc>`:



## Public Types

- typedef `__gnu_cxx::__alloc_traits<_Alloc>::template rebind<_Tp>::other_Tp_alloc_type`
- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__alloc_traits<_Tp_alloc_type>::pointer` **pointer**

## Public Member Functions

- **\_Vector\_base** (const allocator\_type &\_\_a)
- **\_Vector\_base** (size\_t \_\_n)
- **\_Vector\_base** (size\_t \_\_n, const allocator\_type &\_\_a)
- **\_Vector\_base** (\_Tp\_alloc\_type &&\_\_a)
- **\_Vector\_base** (\_Vector\_base &&\_\_x)
- **\_Vector\_base** (\_Vector\_base &&\_\_x, const allocator\_type &\_\_a)
- pointer **\_M\_allocate** (size\_t \_\_n)
- void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- allocator\_type **get\_allocator** () const noexcept

## Public Attributes

- `_Vector_impl_M_impl`

#### 4.563.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::_Vector_base< _Tp, _Alloc >
```

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

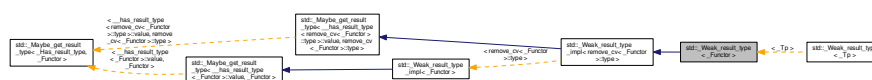
Definition at line 72 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- `std::vector.h`

#### 4.564 `std::_Weak_result_type<_Functor>` Struct Template Reference

Inheritance diagram for `std:: Weak_result_type<_Functor>`:



#### 4.564.1 Detailed Description

```
template<typename _Functor> struct std::Weak_result_type< _Functor >
```

Strip top-level cv-qualifiers from the function object and let `Weak result type impl` perform the real work.

Definition at line 184 of file functional.

The documentation for this struct was generated from the following file:

- functional

#### 4.565 `std::Weak_result_type_impl<_Functor>` Struct Template Reference

Inheritance diagram for `std::_Weak_result_type_impl<_Functor>`:



#### 4.565.1 Detailed Description

```
template<typename _Functor> struct std::Weak_result_type_impl< _Functor >
```

Base class for any function object that has a weak result type, as defined in 3.3/3 of TR1.

Definition at line 86 of file functional.

The documentation for this struct was generated from the following file:

- functional

#### 4.566 `std::Weak_result_type_impl<_Res(&)(_ArgTypes...)>` Struct Template Reference

## Public Types

- typedef \_Res **result\_type**

#### 4.566.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)>
```

Retrieve the result type for a function reference.

Definition at line 125 of file functional.

The documentation for this struct was generated from the following file:

- functional

#### 4.567 `std::Weak_result_type_impl< _Res(*)(_ArgTypes...)>` Struct Template Reference

## Public Types

- typedef Res **result** type

#### 4.567.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>
```

Retrieve the result type for a function pointer.

Definition at line 134 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.568 `std::Weak_result_type_impl< _Res(_ArgTypes...)>` Struct Template Reference

##### Public Types

- typedef `_Res` **result\_type**

##### 4.568.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::Weak_result_type_impl< _Res(_ArgTypes...)>
```

Retrieve the result type for a function type.

Definition at line 92 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.569 `std::Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >` Struct Template Reference

##### Public Types

- typedef `_Res` **result\_type**

##### 4.569.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct std::Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >
```

Retrieve result type for a const member function pointer.

Definition at line 152 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.570 `std::Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >` Struct Template Reference

##### Public Types

- typedef `_Res` **result\_type**

##### 4.570.1 Detailed Description

## 4.571 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >` Struct Template Reference 1488

```
template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >
```

Retrieve result type for a const volatile member function pointer.

Definition at line 170 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.571 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >` Struct Template Reference

### Public Types

- typedef `_Res` **result\_type**

### 4.571.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >
```

Retrieve result type for a volatile member function pointer.

Definition at line 161 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.572 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)>` Struct Template Reference

### Public Types

- typedef `_Res` **result\_type**

### 4.572.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)>
```

Retrieve result type for a member function pointer.

Definition at line 143 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.573 `std::add_const< _Tp >` Struct Template Reference

### Public Types

- typedef `_Tp` const **type**

## 4.573.1 Detailed Description

```
template<typename _Tp>struct std::add_const< _Tp >
```

`add_const`

Definition at line 1353 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.574 `std::add_cv< _Tp >` Struct Template Reference

## Public Types

- typedef `add_const< typename add_volatile< _Tp >::type >::type` **type**

## 4.574.1 Detailed Description

```
template<typename _Tp>struct std::add_cv< _Tp >
```

`add_cv`

Definition at line 1363 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.575 `std::add_lvalue_reference< _Tp >` Struct Template Reference

Inherits `std::__add_lvalue_reference_helper< _Tp, bool, bool >`.

## Public Types

- typedef `_Tp` **type**

## 4.575.1 Detailed Description

```
template<typename _Tp>struct std::add_lvalue_reference< _Tp >
```

`add_lvalue_reference`

Definition at line 1402 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.576 std::add\_pointer< \_Tp > Struct Template Reference

### Public Types

- typedef [remove\\_reference< \\_Tp >](#)  
::type \* **type**

#### 4.576.1 Detailed Description

template<typename \_Tp>struct std::add\_pointer< \_Tp >

add\_pointer

Definition at line 1667 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.577 std::add\_rvalue\_reference< \_Tp > Struct Template Reference

Inherits std::\_\_add\_rvalue\_reference\_helper< \_Tp, bool >.

### Public Types

- typedef \_Tp **type**

#### 4.577.1 Detailed Description

template<typename \_Tp>struct std::add\_rvalue\_reference< \_Tp >

add\_rvalue\_reference

Definition at line 1418 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.578 std::add\_volatile< \_Tp > Struct Template Reference

### Public Types

- typedef \_Tp volatile **type**

#### 4.578.1 Detailed Description

template<typename \_Tp>struct std::add\_volatile< \_Tp >

add\_volatile

Definition at line 1358 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.579 std::adopt\_lock\_t Struct Reference

### 4.579.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.

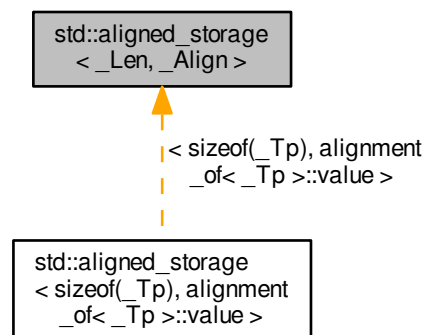
Definition at line 396 of file mutex.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 4.580 std::aligned\_storage< \_Len, \_Align > Struct Template Reference

Inheritance diagram for std::aligned\_storage< \_Len, \_Align >:



### 4.580.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = _alignof_(typename __aligned_storage_msa<_Len>::__type)> struct std::aligned_storage<_Len, _Align>
```

Alignment type.

The value of `_Align` is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than `_Len` (3.9). The member typedef type shall be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

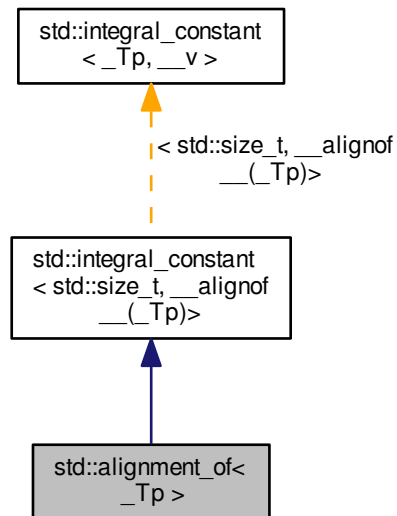
Definition at line 1693 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.581 std::alignment\_of< \_Tp > Struct Template Reference

Inheritance diagram for std::alignment\_of< \_Tp >:



### Public Types

- typedef [integral\\_constant](#) `< std::size_t, __v > type`
- typedef `std::size_t value_type`

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `std::size_t value`

#### 4.581.1 Detailed Description

```
template<typename _Tp> struct std::alignment_of< _Tp >
```

alignment\_of

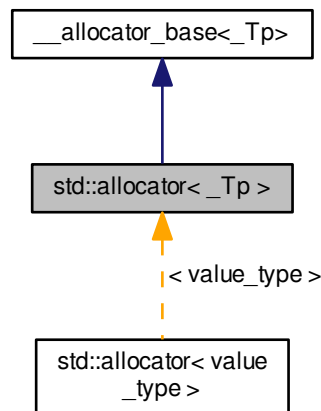
Definition at line 1238 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.582 std::allocator< \_Tp > Struct Template Reference

Inheritance diagram for std::allocator< \_Tp >:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **allocator** (const [allocator](#) &\_\_a) throw ()
- template<typename \_Tp1 >  
  **allocator** (const [allocator](#)< \_Tp1 > &) throw ()
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*=0)

- `template<typename _Up, typename... _Args>`  
`void construct (_Up *__p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type)`
- `template<typename _Up >`  
`void destroy (_Up *__p)`
- `size_type max_size () const noexcept`

#### 4.582.1 Detailed Description

`template<typename _Tp>struct std::allocator< _Tp >`

The *standard* allocator, as per [20.4].

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html> for further details.

#### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

Destroy a range of objects using the supplied allocator. For nondefault allocators we do not optimize away invocation of `destroy()` even if `_Tp` has a trivial destructor.

Definition at line 99 of file `allocator.h`.

The documentation for this struct was generated from the following file:

- [allocator.h](#)

## 4.583 `std::allocator< void >` Class Template Reference

#### Public Types

- `typedef const void * const_pointer`
- `typedef ptrdiff_t difference_type`
- `typedef void * pointer`
- `typedef true\_type propagate_on_container_move_assignment`
- `typedef size_t size_type`
- `typedef void value_type`

#### 4.583.1 Detailed Description

`template<>class std::allocator< void >`

`allocator<void>` specialization.

Definition at line 70 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 4.584 std::allocator\_arg\_t Struct Reference

## 4.584.1 Detailed Description

[allocator.tag]

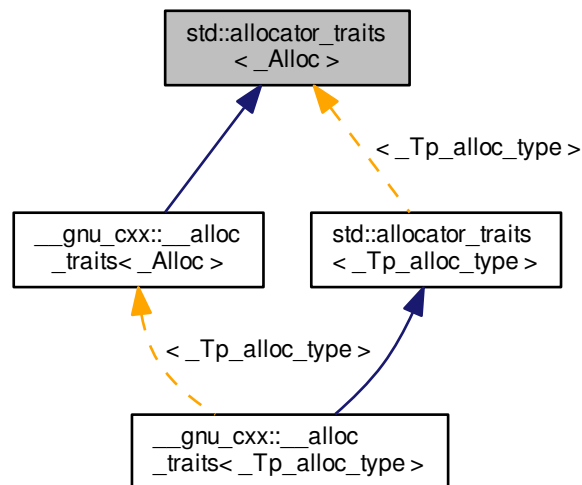
Definition at line 39 of file `uses_allocator.h`.

The documentation for this struct was generated from the following file:

- `uses_allocator.h`

## 4.585 std::allocator\_traits&lt; \_Alloc &gt; Struct Template Reference

Inheritance diagram for `std::allocator_traits< _Alloc >`:



## Public Types

- typedef `_Alloc` [allocator\\_type](#)
- typedef `__const_pointer` [const\\_pointer](#)
- typedef `__const_void_pointer` [const\\_void\\_pointer](#)
- typedef `__difference_type` [difference\\_type](#)
- typedef `__pointer` [pointer](#)
- typedef `__propagate_on_container_copy_assignment` [propagate\\_on\\_container\\_copy\\_assignment](#)
- typedef `__propagate_on_container_move_assignment` [propagate\\_on\\_container\\_move\\_assignment](#)
- typedef `__propagate_on_container_swap` [propagate\\_on\\_container\\_swap](#)

- `template<typename _Tp >`  
`using rebind_alloc = typename __alloctr_rebind<_Alloc, _Tp >::__type`
- `template<typename _Tp >`  
`using rebind_traits = allocator\_traits< rebind\_alloc<_Tp >>`
- `typedef __size_type size\_type`
- `typedef _Alloc::value_type value\_type`
- `typedef __void_pointer void\_pointer`

#### Static Public Member Functions

- static [pointer allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n)
- static [pointer allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- `template<typename _Tp, typename... _Args>`  
`static auto construct (_Alloc &__a, _Tp *__p, _Args &&...__args) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...))`
- static void [deallocate](#) (\_Alloc &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- `template<class _Tp >`  
`static void destroy (_Alloc &__a, _Tp *__p)`
- static [size\\_type max\\_size](#) (const \_Alloc &\_\_a)
- static \_Alloc [select\\_on\\_container\\_copy\\_construction](#) (const \_Alloc &\_\_rhs)

#### 4.585.1 Detailed Description

`template<typename _Alloc> struct std::allocator_traits<_Alloc>`

Uniform interface to all allocator types.

Definition at line 87 of file bits/alloc\_traits.h.

#### 4.585.2 Member Typedef Documentation

##### 4.585.2.1 `template<typename _Alloc> typedef _Alloc std::allocator_traits<_Alloc>::allocator_type`

The allocator type.

Definition at line 90 of file bits/alloc\_traits.h.

##### 4.585.2.2 `template<typename _Alloc> typedef __const_pointer std::allocator_traits<_Alloc>::const_pointer`

The allocator's const pointer type.

`_Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

Definition at line 120 of file bits/alloc\_traits.h.

##### 4.585.2.3 `template<typename _Alloc> typedef __const_void_pointer std::allocator_traits<_Alloc>::const_void_pointer`

The allocator's const void pointer type.

`_Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 142 of file bits/alloc\_traits.h.

**4.585.2.4 template<typename \_Alloc> typedef \_\_difference\_type std::allocator\_traits<\_Alloc>::difference\_type**

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

Definition at line 153 of file `bits/alloc_traits.h`.

**4.585.2.5 template<typename \_Alloc> typedef \_\_pointer std::allocator\_traits<\_Alloc>::pointer**

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

Definition at line 109 of file `bits/alloc_traits.h`.

**4.585.2.6 template<typename \_Alloc> typedef \_\_propagate\_on\_container\_copy\_assignment std::allocator\_traits<\_Alloc>::propagate\_on\_container\_copy\_assignment**

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 176 of file `bits/alloc_traits.h`.

**4.585.2.7 template<typename \_Alloc> typedef \_\_propagate\_on\_container\_move\_assignment std::allocator\_traits<\_Alloc>::propagate\_on\_container\_move\_assignment**

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 188 of file `bits/alloc_traits.h`.

**4.585.2.8 template<typename \_Alloc> typedef \_\_propagate\_on\_container\_swap std::allocator\_traits<\_Alloc>::propagate\_on\_container\_swap**

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 199 of file `bits/alloc_traits.h`.

**4.585.2.9 template<typename \_Alloc> typedef \_\_size\_type std::allocator\_traits<\_Alloc>::size\_type**

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

Definition at line 164 of file `bits/alloc_traits.h`.

**4.585.2.10 template<typename \_Alloc> typedef \_\_value\_type std::allocator\_traits<\_Alloc>::value\_type**

The allocated type.

Definition at line 92 of file `bits/alloc_traits.h`.

**4.585.2.11 template<typename \_Alloc> typedef \_\_void\_pointer std::allocator\_traits<\_Alloc>::void\_pointer**

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 131 of file bits/alloc\_traits.h.

#### 4.585.3 Member Function Documentation

**4.585.3.1** `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate ( _Alloc & __a, size_type __n )  
[inline],[static]`

Allocate memory.

##### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

Definition at line 352 of file bits/alloc\_traits.h.

**4.585.3.2** `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate ( _Alloc & __a, size_type __n,  
const_void_pointer __hint ) [inline],[static]`

Allocate memory.

##### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

##### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 367 of file bits/alloc\_traits.h.

**4.585.3.3** `template<typename _Alloc> template<typename _Tp, typename... _Args> static auto std::allocator_traits<  
_Alloc>::construct ( _Alloc & __a, _Tp * __p, _Args &&... __args )-> decltype(_S_construct(__a, __p,  
std::forward<_Args>(__args)...)) [inline],[static]`

Construct an object of type `_Tp`.

##### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

Definition at line 393 of file bits/alloc\_traits.h.

4.585.3.4 `template<typename _Alloc> static void std::allocator_traits<_Alloc>::deallocate ( _Alloc & __a, pointer __p, size_type __n ) [inline],[static]`

Deallocate memory.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

Definition at line 378 of file `bits/alloc_traits.h`.

4.585.3.5 `template<typename _Alloc> template<class _Tp> static void std::allocator_traits<_Alloc>::destroy ( _Alloc & __a, _Tp* __p ) [inline],[static]`

Destroy an object of type `_Tp`.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__a</code> | An allocator.                    |
| <code>__p</code> | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 406 of file `bits/alloc_traits.h`.

4.585.3.6 `template<typename _Alloc> static size_type std::allocator_traits<_Alloc>::max_size ( const _Alloc & __a ) [inline],[static]`

The maximum supported allocation size.

#### Parameters

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

#### Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 417 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`.

4.585.3.7 `template<typename _Alloc> static _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction ( const _Alloc & __rhs ) [inline],[static]`

Obtain an allocator to use when copying a container.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 429 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

4.586 `std::array<_Tp, _Nm >` Struct Template Reference

## Public Types

- `typedef ::__array_traits<_Tp, _Nm > _AT_Type`
- `typedef const value_type * const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator < const_iterator > const_reverse_iterator`
- `typedef std::ptrdiff_t difference_type`
- `typedef value_type * iterator`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator < iterator > reverse_iterator`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- reference **at** (size\_type \_\_n)
- constexpr const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- constexpr const\_reference **back** () const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- pointer **data** () noexcept
- const\_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- void **fill** (const value\_type &\_\_u)
- reference **front** ()

- constexpr const\_reference **front** () const
- constexpr size\_type **max\_size** () const noexcept
- reference **operator[]** (size\_type \_\_n)
- constexpr const\_reference **operator[]** (size\_type \_\_n) const noexcept
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr size\_type **size** () const noexcept
- void **swap** ([array](#) &\_\_other) noexcept(noexcept(swap(std

#### Public Attributes

- `_AT_Type::Type` **\_M\_elems**

#### 4.586.1 Detailed Description

template<typename \_Tp, std::size\_t \_Nm>struct std::array< \_Tp, \_Nm >

A standard container for storing a fixed size sequence of elements.

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

#### Template Parameters

|           |                                                  |
|-----------|--------------------------------------------------|
| <i>Tp</i> | Type of element. Required to be a complete type. |
| <i>N</i>  | Number of elements.                              |

Definition at line 81 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

## 4.587 std::atomic< \_Tp > Struct Template Reference

#### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** ( \_Tp \_\_i) noexcept
- bool **compare\_exchange\_strong** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_strong** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatilenoexcept
- bool **compare\_exchange\_strong** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatilenoexcept
- bool **compare\_exchange\_weak** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatilenoexcept
- bool **compare\_exchange\_weak** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** ( \_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatilenoexcept

- `_Tp exchange (_Tp __i, memory\_order _m=memory_order_seq_cst)` noexcept
- `_Tp exchange (_Tp __i, memory\_order _m=memory_order_seq_cst)` volatilenoexcept
- `bool is_lock_free ()` const noexcept
- `bool is_lock_free ()` const volatilenoexcept
- `_Tp load (memory\_order _m=memory_order_seq_cst)` const noexcept
- `_Tp load (memory\_order _m=memory_order_seq_cst)` const volatilenoexcept
- `operator _Tp ()` const noexcept
- `operator _Tp ()` const volatilenoexcept
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `_Tp operator= (_Tp __i)` noexcept
- `_Tp operator= (_Tp __i)` volatilenoexcept
- `void store (_Tp __i, memory\_order _m=memory_order_seq_cst)` noexcept
- `void store (_Tp __i, memory\_order _m=memory_order_seq_cst)` volatilenoexcept

#### 4.587.1 Detailed Description

```
template<typename _Tp>struct std::atomic< _Tp >
```

Generic atomic type, primary class template.

#### Template Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>_Tp</code> | Type to be made atomic, must be trivially copyable. |
|------------------|-----------------------------------------------------|

Definition at line 161 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.588 std::atomic< \_Tp \* > Struct Template Reference

#### Public Types

- `typedef \_\_atomic\_base< _Tp * > \_\_base\_type`
- `typedef _Tp * \_\_pointer\_type`

#### Public Member Functions

- `atomic (const atomic &)=delete`
- `constexpr atomic (__pointer_type __p)` noexcept
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory\_order __m1, memory\_order __m2)` noexcept
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory\_order __m1, memory\_order __m2)` volatilenoexcept
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory\_order __m=memory_order_seq_cst)` noexcept
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory\_order __m=memory_order_seq_cst)` volatilenoexcept

- bool **compare\_exchange\_weak** (\_\_pointer\_type &\_\_p1, \_\_pointer\_type \_\_p2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_pointer\_type &\_\_p1, \_\_pointer\_type \_\_p2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) [volatile](#)noexcept
- bool **compare\_exchange\_weak** (\_\_pointer\_type &\_\_p1, \_\_pointer\_type \_\_p2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_pointer\_type &\_\_p1, \_\_pointer\_type \_\_p2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) [volatile](#)noexcept
- \_\_pointer\_type **exchange** (\_\_pointer\_type \_\_p, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_pointer\_type **exchange** (\_\_pointer\_type \_\_p, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) [volatile](#)noexcept
- \_\_pointer\_type **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_pointer\_type **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) [volatile](#)noexcept
- \_\_pointer\_type **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_pointer\_type **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) [volatile](#)noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const [volatile](#)noexcept
- \_\_pointer\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_pointer\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const [volatile](#)noexcept
- **operator** \_\_pointer\_type () const noexcept
- **operator** \_\_pointer\_type () const [volatile](#)noexcept
- \_\_pointer\_type **operator++** (int) noexcept
- \_\_pointer\_type **operator++** (int) [volatile](#)noexcept
- \_\_pointer\_type **operator++** () noexcept
- \_\_pointer\_type **operator++** () [volatile](#)noexcept
- \_\_pointer\_type **operator+=** (ptrdiff\_t \_\_d) noexcept
- \_\_pointer\_type **operator+=** (ptrdiff\_t \_\_d) [volatile](#)noexcept
- \_\_pointer\_type **operator--** (int) noexcept
- \_\_pointer\_type **operator--** (int) [volatile](#)noexcept
- \_\_pointer\_type **operator--** () noexcept
- \_\_pointer\_type **operator--** () [volatile](#)noexcept
- \_\_pointer\_type **operator-=** (ptrdiff\_t \_\_d) noexcept
- \_\_pointer\_type **operator-=** (ptrdiff\_t \_\_d) [volatile](#)noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) [volatile](#)=delete
- \_\_pointer\_type **operator=** (\_\_pointer\_type \_\_p) noexcept
- \_\_pointer\_type **operator=** (\_\_pointer\_type \_\_p) [volatile](#)noexcept
- void **store** (\_\_pointer\_type \_\_p, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_pointer\_type \_\_p, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) [volatile](#)noexcept

#### Public Attributes

- [\\_\\_base\\_type](#) [\\_M\\_b](#)

#### 4.588.1 Detailed Description

template<typename \_Tp>struct std::atomic<\_Tp \* >

Partial specialization for pointer types.

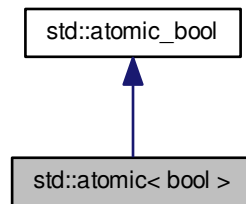
Definition at line 290 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.589 std::atomic&lt; bool &gt; Struct Template Reference

Inheritance diagram for std::atomic< bool >:



## Public Types

- typedef [atomic\\_bool](#) **\_\_base\_type**
- typedef bool **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** ([\\_\\_integral\\_type](#) \_\_i) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile-noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile-noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile-noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile-noexcept
- **operator bool** () const noexcept
- **operator bool** () const volatile-noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept

## 4.589.1 Detailed Description

```
template<> struct std::atomic< bool >
```

Explicit specialization for bool.

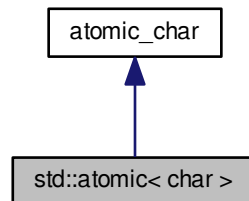
Definition at line 480 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.590 `std::atomic< char >` Struct Template Reference

Inheritance diagram for `std::atomic< char >`:



## Public Types

- typedef [atomic\\_char](#) `__base_type`
- typedef char `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.590.1 Detailed Description

`template<> struct std::atomic< char >`

Explicit specialization for char.

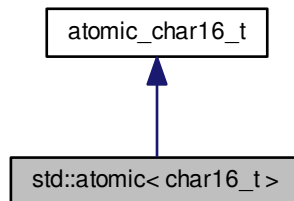
Definition at line 499 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.591 `std::atomic< char16_t >` Struct Template Reference

Inheritance diagram for `std::atomic< char16_t >`:



## Public Types

- typedef [atomic\\_char16\\_t](#) `__base_type`
- typedef `char16_t` `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.591.1 Detailed Description

template<> struct std::atomic< char16\_t >

Explicit specialization for char16\_t.

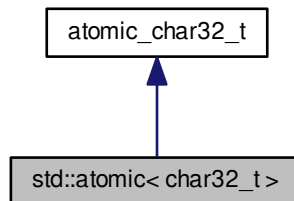
Definition at line 727 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.592 std::atomic&lt; char32\_t &gt; Struct Template Reference

Inheritance diagram for std::atomic< char32\_t >:



## Public Types

- typedef [atomic\\_char32\\_t](#) \_\_base\_type
- typedef char32\_t \_\_integral\_type

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.592.1 Detailed Description

```
template<> struct std::atomic< char32_t >
```

Explicit specialization for `char32_t`.

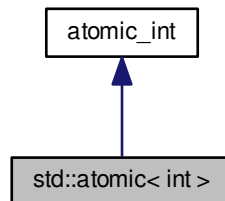
Definition at line 746 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.593 `std::atomic< int >` Struct Template Reference

Inheritance diagram for `std::atomic< int >`:



## Public Types

- typedef [atomic\\_int](#) `__base_type`
- typedef int `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) `volatile`noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) `volatile`noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) `volatile`noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.593.1 Detailed Description

```
template<> struct std::atomic< int >
```

Explicit specialization for int.

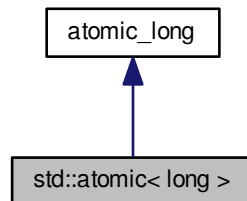
Definition at line 594 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.594 `std::atomic< long >` Struct Template Reference

Inheritance diagram for `std::atomic< long >`:



## Public Types

- typedef [atomic\\_long](#) `__base_type`
- typedef long `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) `volatile`noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) `volatile`noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) `volatile`noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.594.1 Detailed Description

`template<> struct std::atomic< long >`

Explicit specialization for long.

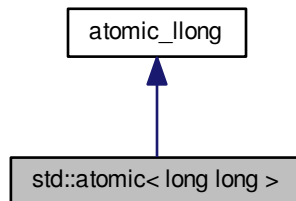
Definition at line 632 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.595 `std::atomic< long long >` Struct Template Reference

Inheritance diagram for `std::atomic< long long >`:



## Public Types

- typedef [atomic\\_llong](#) `__base_type`
- typedef long long `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) `volatile`noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) `volatile`noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) `volatile`noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.595.1 Detailed Description

`template<> struct std::atomic< long long >`

Explicit specialization for long long.

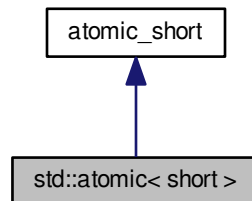
Definition at line 670 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.596 `std::atomic< short >` Struct Template Reference

Inheritance diagram for `std::atomic< short >`:



## Public Types

- typedef [atomic\\_short](#) `__base_type`
- typedef short `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.596.1 Detailed Description

`template<> struct std::atomic< short >`

Explicit specialization for short.

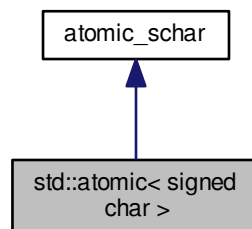
Definition at line 556 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.597 `std::atomic< signed char >` Struct Template Reference

Inheritance diagram for `std::atomic< signed char >`:



## Public Types

- typedef [atomic\\_schar](#) **\_\_base\_type**
- typedef signed char **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.597.1 Detailed Description

`template<> struct std::atomic< signed char >`

Explicit specialization for signed char.

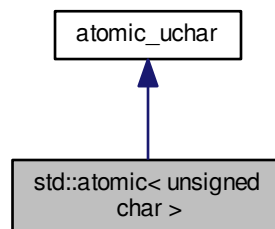
Definition at line 518 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.598 `std::atomic< unsigned char >` Struct Template Reference

Inheritance diagram for `std::atomic< unsigned char >`:



## Public Types

- typedef [atomic\\_uchar](#) **\_\_base\_type**
- typedef unsigned char **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.598.1 Detailed Description

template<>struct std::atomic< unsigned char >

Explicit specialization for unsigned char.

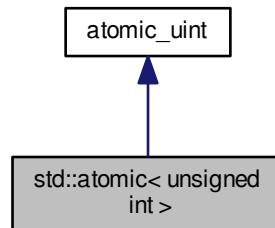
Definition at line 537 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.599 std::atomic&lt; unsigned int &gt; Struct Template Reference

Inheritance diagram for std::atomic< unsigned int >:



## Public Types

- typedef [atomic\\_uint](#) **\_\_base\_type**
- typedef unsigned int **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.599.1 Detailed Description

template<>struct std::atomic< unsigned int >

Explicit specialization for unsigned int.

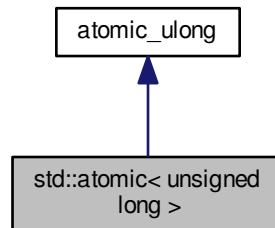
Definition at line 613 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.600 std::atomic&lt; unsigned long &gt; Struct Template Reference

Inheritance diagram for std::atomic< unsigned long >:



## Public Types

- typedef [atomic\\_ulong](#) **\_\_base\_type**
- typedef unsigned long **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.600.1 Detailed Description

template<>struct std::atomic< unsigned long >

Explicit specialization for unsigned long.

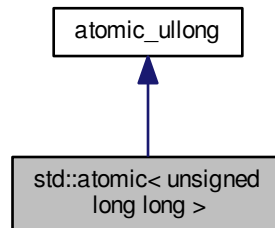
Definition at line 651 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.601 std::atomic&lt; unsigned long long &gt; Struct Template Reference

Inheritance diagram for std::atomic< unsigned long long >:



## Public Types

- typedef [atomic\\_ullong](#) **\_\_base\_type**
- typedef unsigned long long **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.601.1 Detailed Description

template<>struct std::atomic< unsigned long long >

Explicit specialization for unsigned long long.

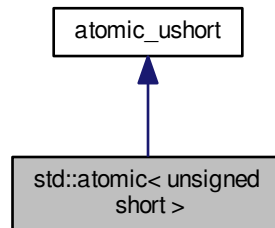
Definition at line 689 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.602 std::atomic&lt; unsigned short &gt; Struct Template Reference

Inheritance diagram for std::atomic< unsigned short >:



## Public Types

- typedef [atomic\\_ushort](#) **\_\_base\_type**
- typedef unsigned short **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.602.1 Detailed Description

`template<> struct std::atomic< unsigned short >`

Explicit specialization for unsigned short.

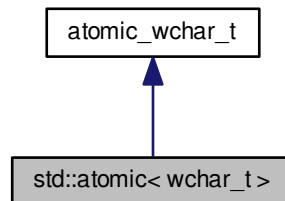
Definition at line 575 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.603 `std::atomic< wchar_t >` Struct Template Reference

Inheritance diagram for `std::atomic< wchar_t >`:



## Public Types

- typedef [atomic\\_wchar\\_t](#) `__base_type`
- typedef `wchar_t` `__integral_type`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (`__int_type` &\_\_i1, `__int_type` \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept

- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator** \_\_int\_type () const noexcept
- **operator** \_\_int\_type () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

## 4.603.1 Detailed Description

```
template<> struct std::atomic< wchar_t >
```

Explicit specialization for wchar\_t.

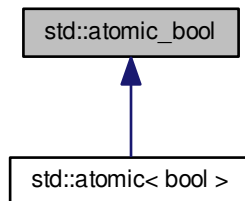
Definition at line 708 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.604 std::atomic\_bool Struct Reference

Inheritance diagram for std::atomic\_bool:



## Public Member Functions

- **atomic\_bool** (const [atomic\\_bool](#) &)=delete
- constexpr **atomic\_bool** (bool \_\_i) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile-noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile-noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile-noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile-noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept

- `bool load (memory_order __m=memory_order_seq_cst) const` `volatile` `noexcept`
- `operator bool ()` `const` `noexcept`
- `operator bool ()` `const` `volatile` `noexcept`
- `atomic_bool & operator= (const atomic_bool &)=delete`
- `atomic_bool & operator= (const atomic_bool &) volatile=delete`
- `bool operator= (bool __i) noexcept`
- `bool operator= (bool __i) volatile` `noexcept`
- `void store (bool __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (bool __i, memory_order __m=memory_order_seq_cst) volatile` `noexcept`

#### 4.604.1 Detailed Description

`atomic_bool`

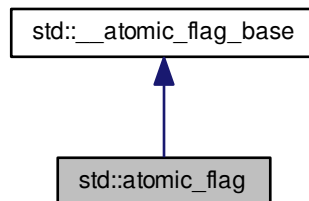
Definition at line 54 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.605 std::atomic\_flag Struct Reference

Inheritance diagram for `std::atomic_flag`:



#### Public Member Functions

- `atomic_flag (const atomic_flag &)=delete`
- `constexpr atomic_flag (bool __i) noexcept`
- `void clear (memory_order __m=memory_order_seq_cst) noexcept`
- `void clear (memory_order __m=memory_order_seq_cst) volatile` `noexcept`
- `atomic_flag & operator= (const atomic_flag &)=delete`
- `atomic_flag & operator= (const atomic_flag &) volatile=delete`
- `bool test_and_set (memory_order __m=memory_order_seq_cst) noexcept`
- `bool test_and_set (memory_order __m=memory_order_seq_cst) volatile` `noexcept`

## Public Attributes

- `__atomic_flag_data_type _M_i`

## 4.605.1 Detailed Description

`atomic_flag`

Definition at line 271 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 4.606 std::auto\_ptr&lt; \_Tp &gt; Class Template Reference

## Public Types

- `typedef _Tp element_type`

## Public Member Functions

- `auto_ptr (element_type *__p=0) throw ()`
- `auto_ptr (auto_ptr &__a) throw ()`
- `template<typename _Tp1 >  
auto_ptr (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr (auto_ptr_ref< element_type > &__ref) throw ()`
- `~auto_ptr ()`
- `element_type * get () const throw ()`
- `template<typename _Tp1 >  
operator auto_ptr< _Tp1 > () throw ()`
- `template<typename _Tp1 >  
operator auto_ptr_ref< _Tp1 > () throw ()`
- `element_type & operator* () const throw ()`
- `element_type * operator-> () const throw ()`
- `auto_ptr & operator= (auto_ptr &__a) throw ()`
- `template<typename _Tp1 >  
auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr & operator= (auto_ptr_ref< element_type > &__ref) throw ()`
- `element_type * release () throw ()`
- `void reset (element_type *__p=0) throw ()`

## 4.606.1 Detailed Description

```
template<typename _Tp>class std::auto_ptr< _Tp >
```

A simple smart pointer providing strict ownership semantics.

The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS 127. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 87 of file `auto_ptr.h`.

#### 4.606.2 Member Typedef Documentation

##### 4.606.2.1 `template<typename _Tp> typedef _Tp std::auto_ptr<_Tp>::element_type`

The pointed-to type.

Definition at line 94 of file `auto_ptr.h`.

#### 4.606.3 Constructor & Destructor Documentation

##### 4.606.3.1 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( element_type * __p = 0 ) throw () [inline], [explicit]`

An `auto_ptr` is usually constructed from a raw pointer.

###### Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>__p</code> | A pointer (defaults to NULL). |
|------------------|-------------------------------|

This object now *owns* the object pointed to by `__p`.

Definition at line 103 of file `auto_ptr.h`.

##### 4.606.3.2 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( auto_ptr<_Tp> & __a ) throw () [inline]`

An `auto_ptr` can be constructed from another `auto_ptr`.

###### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of the same type. |
|------------------|-------------------------------------------------|

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 112 of file `auto_ptr.h`.

4.606.3.3 `template<typename _Tp> template<typename _Tp1 > std::auto_ptr< _Tp >::auto_ptr ( auto_ptr< _Tp1 > & __a ) throw () [inline]`

An auto\_ptr can be constructed from another auto\_ptr.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__a</code> | Another auto_ptr of a different but related type. |
|------------------|---------------------------------------------------|

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 125 of file auto\_ptr.h.

4.606.3.4 `template<typename _Tp> std::auto_ptr< _Tp >::~~auto_ptr ( ) [inline]`

When the auto\_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 170 of file auto\_ptr.h.

4.606.3.5 `template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr ( auto_ptr_ref< element_type > __ref ) throw () [inline]`

Automatic conversions.

These operations convert an auto\_ptr into and from an auto\_ptr\_ref automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(....);
```

Definition at line 260 of file auto\_ptr.h.

## 4.606.4 Member Function Documentation

4.606.4.1 `template<typename _Tp> element_type* std::auto_ptr< _Tp >::get ( void ) const throw () [inline]`

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

#### Note

This auto\_ptr still owns the memory.

Definition at line 211 of file auto\_ptr.h.

4.606.4.2 `template<typename _Tp> element_type& std::auto_ptr< _Tp >::operator* ( ) const throw () [inline]`

Smart pointer dereferencing.

If this `auto_ptr` no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 181 of file `auto_ptr.h`.

4.606.4.3 `template<typename _Tp> element_type* std::auto_ptr< _Tp >::operator-> ( ) const throw () [inline]`

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 194 of file `auto_ptr.h`.

4.606.4.4 `template<typename _Tp> auto_ptr& std::auto_ptr< _Tp >::operator= ( auto_ptr< _Tp > & __a ) throw () [inline]`

`auto_ptr` assignment operator.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of the same type. |
|------------------|-------------------------------------------------|

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 136 of file `auto_ptr.h`.

References `std::auto_ptr< _Tp >::reset()`.

4.606.4.5 `template<typename _Tp> template<typename Tp1 > auto_ptr& std::auto_ptr< _Tp >::operator= ( auto_ptr< Tp1 > & __a ) throw () [inline]`

`auto_ptr` assignment operator.

#### Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of a different but related type. |
|------------------|----------------------------------------------------------------|

A pointer-to-`Tp1` must be convertible to a pointer-to-`Tp/element_type`.

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 154 of file `auto_ptr.h`.

References `std::auto_ptr< _Tp >::reset()`.

4.606.4.6 `template<typename _Tp> element_type* std::auto_ptr< _Tp >::release ( ) throw () [inline]`

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

**Note**

This auto\_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 225 of file auto\_ptr.h.

4.606.4.7 `template<typename _Tp> void std::auto_ptr< _Tp >::reset ( element_type * __p = 0 ) throw ()` `[inline]`

Forcibly deletes the managed object.

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <code>__p</code> | A pointer (defaults to NULL). |
|------------------|-------------------------------|

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.

Definition at line 240 of file auto\_ptr.h.

Referenced by `std::auto_ptr< _Tp >::operator=()`.

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

**4.607 std::auto\_ptr\_ref< \_Tp1 > Struct Template Reference****Public Member Functions**

- **auto\_ptr\_ref** (`_Tp1 * __p`)

**Public Attributes**

- `_Tp1 * _M_ptr`

**4.607.1 Detailed Description**

`template<typename _Tp1> struct std::auto_ptr_ref< _Tp1 >`

A wrapper class to provide auto\_ptr with reference semantics. For example, an auto\_ptr can be assigned (or constructed from) the result of a function which returns an auto\_ptr by value.

All the auto\_ptr\_ref stuff should happen behind the scenes.

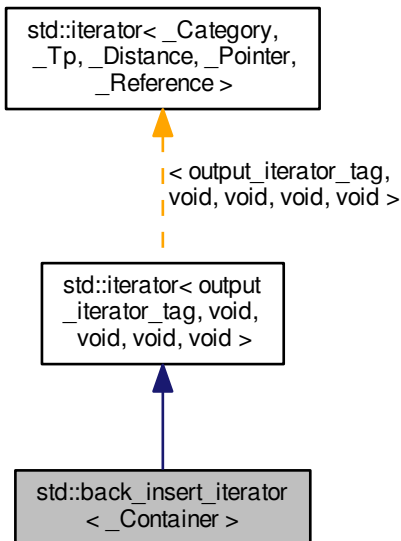
Definition at line 48 of file auto\_ptr.h.

The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

## 4.608 std::back\_insert\_iterator&lt; \_Container &gt; Class Template Reference

Inheritance diagram for std::back\_insert\_iterator< \_Container >:



## Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

## Public Member Functions

- `back_insert_iterator` (`_Container &__x`)
- `back_insert_iterator` & `operator*` ()
- `back_insert_iterator` & `operator++` ()
- `back_insert_iterator` `operator++` (int)
- `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container *` `container`

#### 4.608.1 Detailed Description

```
template<typename _Container>class std::back_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using push\_back.

Tip: Using the back\_inserter function to create these iterators can save typing.

Definition at line 402 of file stl\_iterator.h.

#### 4.608.2 Member Typedef Documentation

4.608.2.1 `template<typename _Container> typedef _Container std::back_insert_iterator< _Container >::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 410 of file stl\_iterator.h.

4.608.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type` [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

4.608.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

4.608.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

4.608.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference` [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

4.608.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

#### 4.608.3 Constructor & Destructor Documentation

4.608.3.1 `template<typename _Container> std::back_insert_iterator< _Container >::back_insert_iterator ( _Container & _x )` [inline],[explicit]

The only way to create this iterator is with a container.

Definition at line 414 of file stl\_iterator.h.

#### 4.608.4 Member Function Documentation

4.608.4.1 `template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator* ( ) [inline]`

Simply returns *\*this*.

Definition at line 452 of file stl\_iterator.h.

4.608.4.2 `template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator++ ( ) [inline]`

Simply returns *\*this*. (This iterator does not *move*.)

Definition at line 457 of file stl\_iterator.h.

4.608.4.3 `template<typename _Container> back_insert_iterator std::back_insert_iterator<_Container>::operator++ ( int ) [inline]`

Simply returns *\*this*. (This iterator does not *move*.)

Definition at line 462 of file stl\_iterator.h.

4.608.4.4 `template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator= ( const typename _Container::value_type & __value ) [inline]`

#### Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

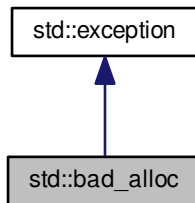
Definition at line 436 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.609 `std::bad_alloc` Class Reference

Inheritance diagram for `std::bad_alloc`:



### Public Member Functions

- virtual const char \* [what](#) () const throw ()

#### 4.609.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 54 of file `new`.

#### 4.609.2 Member Function Documentation

##### 4.609.2.1 virtual const char\* `std::bad_alloc::what` ( ) const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

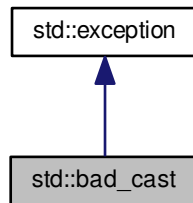
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [new](#)

## 4.610 `std::bad_cast` Class Reference

Inheritance diagram for `std::bad_cast`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 4.610.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 187 of file `typeinfo`.

#### 4.610.2 Member Function Documentation

##### 4.610.2.1 virtual const char\* `std::bad_cast::what` ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

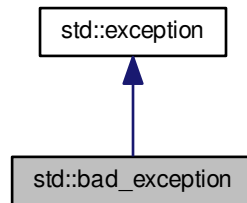
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 4.611 `std::bad_exception` Class Reference

Inheritance diagram for `std::bad_exception`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 4.611.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 73 of file `exception`.

#### 4.611.2 Member Function Documentation

##### 4.611.2.1 virtual const char\* `std::bad_exception::what` ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

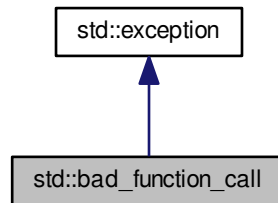
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [exception](#)

## 4.612 `std::bad_function_call` Class Reference

Inheritance diagram for `std::bad_function_call`:



### Public Member Functions

- `const char * what () const noexcept`

#### 4.612.1 Detailed Description

Exception class thrown when class template function's `operator()` is called with an empty target.

Definition at line 1767 of file `functional`.

#### 4.612.2 Member Function Documentation

##### 4.612.2.1 `const char* std::bad_function_call::what ( ) const` `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error.

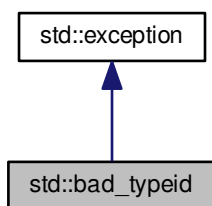
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [functional](#)

## 4.613 `std::bad_typeid` Class Reference

Inheritance diagram for `std::bad_typeid`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 4.613.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 204 of file `typeinfo`.

#### 4.613.2 Member Function Documentation

**4.613.2.1** virtual const char\* `std::bad_typeid::what ( ) const` [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

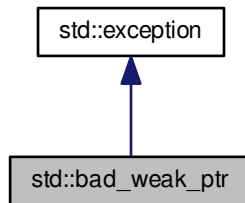
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 4.614 `std::bad_weak_ptr` Class Reference

Inheritance diagram for `std::bad_weak_ptr`:



### Public Member Functions

- virtual char const \* [what](#) () const noexcept

#### 4.614.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

Definition at line 64 of file `shared_ptr_base.h`.

#### 4.614.2 Member Function Documentation

##### 4.614.2.1 virtual char const\* `std::bad_weak_ptr::what` ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

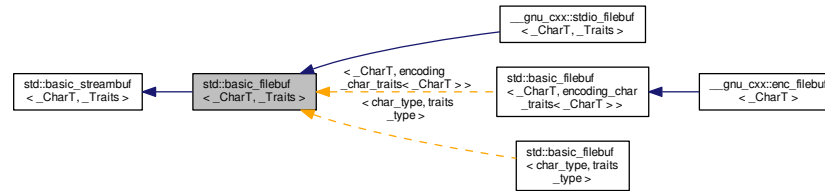
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 4.615 std::basic\_filebuf&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_filebuf< \_CharT, \_Traits >:



## Public Types

- typedef `codecvt`< char\_type, char, \_\_state\_type > **\_\_codecvt\_type**
- typedef `__basic_file`< char > **\_\_file\_type**
- typedef `basic_filebuf`< char\_type, traits\_type > **\_\_filebuf\_type**
- typedef traits\_type::state\_type **\_\_state\_type**
- typedef `basic_streambuf`< char\_type, traits\_type > **\_\_streambuf\_type**
- typedef `_CharT` **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef `_Traits` **traits\_type**

## Public Member Functions

- `basic_filebuf` ()
- virtual `~basic_filebuf` ()
- `__filebuf_type` \* `close` ()
- `locale` `getloc` () const
- `streamsize` `in_avail` ()
- bool `is_open` () const throw ()
- `__filebuf_type` \* `open` (const char \* \_\_s, `ios_base::openmode` \_\_mode)
- `__filebuf_type` \* `open` (const `std::string` & \_\_s, `ios_base::openmode` \_\_mode)
- `locale` `pubimbue` (const `locale` & \_\_loc)
- int\_type `sbumpc` ()
- int\_type `sgetc` ()
- `streamsize` `sgetn` (char\_type \* \_\_s, `streamsize` \_\_n)
- int\_type `snextc` ()
- int\_type `sputbackc` (char\_type \_\_c)
- int\_type `sputc` (char\_type \_\_c)
- `streamsize` `sputn` (const char\_type \* \_\_s, `streamsize` \_\_n)
- int\_type `sungetc` ()

- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \*\_\_s, streamsize \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int [pubsync](#) ()

#### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) (streamsize \_\_n)
  - void [\\_\\_safe\\_pbump](#) (streamsize \_\_n)
  - void [\\_M\\_allocate\\_internal\\_buffer](#) ()
  - bool [\\_M\\_convert\\_to\\_external](#) (char\_type \*, streamsize)
  - void [\\_M\\_create\\_pback](#) ()
  - void [\\_M\\_destroy\\_internal\\_buffer](#) () throw ()
  - void [\\_M\\_destroy\\_pback](#) () throw ()
  - int [\\_M\\_get\\_ext\\_pos](#) (\_\_state\_type &\_\_state)
  - pos\_type [\\_M\\_seek](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
  - void [\\_M\\_set\\_buffer](#) (streamsize \_\_off)
  - bool [\\_M\\_terminate\\_output](#) ()
  - void [gbump](#) (int \_\_n)
  - virtual void [imbue](#) (const locale &\_\_loc)
  - virtual int\_type [overflow](#) (int\_type \_\_c=\_Traits::eof())
  - virtual int\_type [pbackfail](#) (int\_type \_\_c=\_Traits::eof())
  - void [pbump](#) (int \_\_n)
  - virtual pos\_type [seekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual pos\_type [seekpos](#) (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) (char\_type \*\_\_s, streamsize \_\_n)
  - void [setg](#) (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
  - void [setp](#) (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
  - virtual [streamsize showmanyc](#) ()
  - virtual int [sync](#) ()
  - virtual int\_type [uflow](#) ()
  - virtual int\_type [underflow](#) ()
  - virtual [streamsize xsgetn](#) (char\_type \*\_\_s, streamsize \_\_n)
  - virtual [streamsize xsputn](#) (const char\_type \*\_\_s, streamsize \_\_n)
- 
- char\_type \* [eback](#) () const
  - char\_type \* [gptr](#) () const
  - char\_type \* [egptr](#) () const
- 
- char\_type \* [pbase](#) () const
  - char\_type \* [pptr](#) () const
  - char\_type \* [epptr](#) () const

## Protected Attributes

- char\_type \* [\\_M\\_buf](#)
  - bool [\\_M\\_buf\\_allocated](#)
  - locale [\\_M\\_buf\\_locale](#)
  - size\_t [\\_M\\_buf\\_size](#)
  - const [\\_\\_codecvt\\_type](#) \* [\\_M\\_codecvt](#)
  - char \* [\\_M\\_ext\\_buf](#)
  - streamsize [\\_M\\_ext\\_buf\\_size](#)
  - char \* [\\_M\\_ext\\_end](#)
  - const char \* [\\_M\\_ext\\_next](#)
  - [\\_\\_file\\_type](#) [\\_M\\_file](#)
  - char\_type \* [\\_M\\_in\\_beg](#)
  - char\_type \* [\\_M\\_in\\_cur](#)
  - char\_type \* [\\_M\\_in\\_end](#)
  - [\\_\\_c\\_lock](#) [\\_M\\_lock](#)
  - [ios\\_base::openmode](#) [\\_M\\_mode](#)
  - char\_type \* [\\_M\\_out\\_beg](#)
  - char\_type \* [\\_M\\_out\\_cur](#)
  - char\_type \* [\\_M\\_out\\_end](#)
  - bool [\\_M\\_reading](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_beg](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_cur](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_last](#)
  - bool [\\_M\\_writing](#)
- 
- char\_type [\\_M\\_pback](#)
  - char\_type \* [\\_M\\_pback\\_cur\\_save](#)
  - char\_type \* [\\_M\\_pback\\_end\\_save](#)
  - bool [\\_M\\_pback\\_init](#)

## Friends

- class [ios\\_base](#)

## 4.615.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_filebuf< _CharT, _Traits >
```

The actual work of input and output (for files).

## Template Parameters

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                          |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> . |

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Requirements on `traits_type`, specific to this class:

- traits\_type::pos\_type must be fpos<traits\_type::state\_type>
- traits\_type::off\_type must be streamoff
- traits\_type::state\_type must be Assignable and DefaultConstructible,
- traits\_type::state\_type() must be the initial state for codecvt.

Definition at line 72 of file fstream.

#### 4.615.2 Constructor & Destructor Documentation

##### 4.615.2.1 template<typename \_CharT, typename \_Traits > std::basic\_filebuf< \_CharT, \_Traits >::basic\_filebuf ( )

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 79 of file fstream.tcc.

References std::basic\_streambuf< \_CharT, \_Traits >::\_M\_buf\_locale.

##### 4.615.2.2 template<typename \_CharT, typename \_Traits> virtual std::basic\_filebuf< \_CharT, \_Traits >::~basic\_filebuf ( ) [inline], [virtual]

The destructor closes the file first.

Definition at line 219 of file fstream.

#### 4.615.3 Member Function Documentation

##### 4.615.3.1 template<typename \_CharT, typename \_Traits> void std::basic\_filebuf< \_CharT, \_Traits >::\_M\_create\_pback ( ) [inline], [protected]

Initializes pback buffers, and moves normal buffers to safety. Assumptions: \_M\_in\_cur has already been moved back

Definition at line 177 of file fstream.

##### 4.615.3.2 template<typename \_CharT, typename \_Traits> void std::basic\_filebuf< \_CharT, \_Traits >::\_M\_destroy\_pback ( ) throw () [inline], [protected]

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 194 of file fstream.

##### 4.615.3.3 template<typename \_CharT, typename \_Traits> void std::basic\_filebuf< \_CharT, \_Traits >::\_M\_set\_buffer ( streamsize \_\_off ) [inline], [protected]

This function sets the pointers of the internal buffer, both get and put areas. Typically:

\_\_off == egptr() - eback() upon underflow/uflow (**read** mode); \_\_off == 0 upon overflow (**write** mode); \_\_off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: epptr() - pbase() == \_M\_buf\_size - 1, since \_M\_buf\_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 395 of file fstream.

4.615.3.4 `template<typename _CharT, typename _Traits> basic_filebuf< _CharT, _Traits >::__filebuf_type *  
std::basic_filebuf< _CharT, _Traits >::close ( )`

Closes the currently associated file.

#### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file `fstream.tcc`.

Referenced by `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, and `std::basic_filebuf< char_type, traits_type >::~~basic_filebuf()`.

4.615.3.5 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eback ( )  
const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 482 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, and `std::basic_streambuf< char_type, traits_type >::sungetc()`.

4.615.3.6 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( )  
const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 488 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

4.615.3.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eptr ( )`  
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 535 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputc()`.

4.615.3.8 `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::gbump ( int __n )`  
`[inline], [protected], [inherited]`

Moving the read position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Definition at line 498 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, and `std::basic_streambuf< char_type, traits_type >::uflow()`.

4.615.3.9 `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const`  
`[inline], [inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

4.615.3.10 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( )`  
`const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence

- `egptr()` returns the end pointer for the input sequence

Definition at line 485 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, and `std::basic_streambuf< char_type, traits_type >::uflow()`.

**4.615.3.11** `template<typename _CharT, typename _Traits> void std::basic_filebuf< _CharT, _Traits >::imbue ( const locale & __loc ) [protected], [virtual]`

Changes translations.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 910 of file `fstream.tcc`.

References `std::ios_base::cur`.

**4.615.3.12** `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline], [inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file `streambuf`.

**4.615.3.13** `template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::is_open ( ) const throw () [inline]`

Returns true if the external file is open.

Definition at line 227 of file `fstream`.

Referenced by `std::basic_ifstream< _CharT, _Traits >::is_open()`, `std::basic_ofstream< _CharT, _Traits >::is_open()`, and `std::basic_fstream< _CharT, _Traits >::is_open()`.

**4.615.3.14** `template<typename _CharT, typename _Traits> basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open ( const char * __s, ios_base::openmode __mode )`

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

| ios_base Flag | combination | stdio equivalent |
|---------------|-------------|------------------|
| +             | ios_base    | Flag             |
| +             | + w         | ++ a             |
| +             | + a         | + a              |
| +             | + w         | + r              |
| +             | + r         | ++ r+            |
| +             | ++ w+       | ++ + a+          |
| +             | ++ a+       | ++ a+            |
| +             | ++ wb       | ++ + ab          |
| +             | ++ ab       | ++ ab            |
| +             | ++ wb       | ++ rb            |
| +             | ++ rb       | ++ + r+b         |
| +             | ++ + w+b    | ++ + + a+b       |
| +             | ++ + a+b    | ++ + + a+b       |

Definition at line 94 of file fstream.tcc.

References std::ios\_base::ate, std::ios\_base::end, and std::basic\_filebuf< \_CharT, \_Traits >::open().

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), and std::basic\_fstream< \_CharT, \_Traits >::open().

**4.615.3.15** template<typename \_CharT, typename \_Traits> \_\_filebuf\_type\* std::basic\_filebuf< \_CharT, \_Traits >::open (const std::string & \_\_s, ios\_base::openmode \_\_mode) [inline]

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, NULL on failure

Definition at line 280 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::open().

**4.615.3.16** template<typename \_CharT, typename \_Traits> basic\_filebuf< \_CharT, \_Traits >::int\_type std::basic\_filebuf< \_CharT, \_Traits >::overflow (int\_type \_\_c = \_Traits::eof()) [protected], [virtual]

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

eof() to indicate failure, something else (usually \_\_c, or not\_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if \_\_c is not eof().

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns eof().

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 422 of file fstream.tcc.

References std::ios\_base::cur, and std::ios\_base::out.

**4.615.3.17** `template<typename _CharT, typename _Traits> basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail( int_type __c = _Traits::eof() ) [protected], [virtual]`

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

eof() on failure, *some other value* on success

**Postcondition**

The constraints of gptr(), eback(), and pptr() are the same as for underflow().

**Note**

Base class version does nothing, returns eof().

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 363 of file fstream.tcc.

References std::ios\_base::cur, and std::ios\_base::in.

**4.615.3.18** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pbase( ) const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence

- `eptr()` returns the end pointer for the output sequence

Definition at line 529 of file `streambuf`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

**4.615.3.19** `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::pbump ( int __n )`  
`[inline], [protected], [inherited]`

Moving the write position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Definition at line 545 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputc()`.

**4.615.3.20** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( )`  
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 532 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputc()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

**4.615.3.21** `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale & __loc )`  
`[inline], [inherited]`

Entry point for `imbue()`.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 209 of file streambuf.

```
4.615.3.22 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out)
 [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

```
4.615.3.23 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos
(pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline],
[inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

```
4.615.3.24 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits
>::pubsetbuf (char_type * __s, streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

```
4.615.3.25 template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ()
[inline], [inherited]
```

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

4.615.3.26 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( )`  
`[inline], [inherited]`

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::istreambuf_iterator< _CharT, _Traits >::operator++()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

4.615.3.27 `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::pos_type`  
`std::basic_filebuf< _CharT, _Traits >::seekoff ( off_type, ios_base::seekdir, ios_base::openmode =`  
`ios_base::in | ios_base::out ) [protected], [virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 713 of file `fstream.tcc`.

References `std::ios_base::cur`.

4.615.3.28 `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf<`  
`_CharT, _Traits >::seekpos ( pos_type, ios_base::openmode = ios_base::in | ios_base::out )`  
`[protected], [virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 773 of file `fstream.tcc`.

References `std::ios_base::beg`.

4.615.3.29 `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::__streambuf_type *`  
`std::basic_filebuf< _CharT, _Traits >::setbuf ( char_type * __s, streamsize __n ) [protected],`  
`[virtual]`

Manipulates the buffer.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__s</code> | Pointer to a buffer area.  |
| <code>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.-html> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 684 of file fstream.tcc.

4.615.3.30 `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setg ( char_type * __gbeg, char_type * __gnext, char_type * __gend )` [inline], [protected], [inherited]

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 509 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

4.615.3.31 `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setp ( char_type * __pbeg, char_type * __pend )` [inline], [protected], [inherited]

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 555 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**4.615.3.32** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( )`  
`[inline], [inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**4.615.3.33** `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (`  
`char_type * __s, streamsize __n ) [inline], [inherited]`

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

**4.615.3.34** `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc (`  
`) [protected], [virtual]`

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1*

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 178 of file `fstream.tcc`.

References `std::ios_base::binary`, and `std::ios_base::in`.

**4.615.3.35** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::snextc ( )`  
`[inline], [inherited]`

Getting the next character.

#### Returns

The next character, or `eof`.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.615.3.36** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (`  
`char_type __c ) [inline], [inherited]`

Pushing characters back into the input stream.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

#### Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 372 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**4.615.3.37** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (`  
`char_type __c ) [inline], [inherited]`

Entry point for all single-character output functions.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

#### Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(-__c)`.

Definition at line 424 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**4.615.3.38** `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn ( const char_type * __s, streamsize __n ) [inline],[inherited]`

Entry point for all single-character output functions.

#### Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 450 of file `streambuf`.

**4.615.3.39** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline],[inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 397 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::unget()`.

**4.615.3.40** `template<typename _CharT, typename _Traits > int std::basic_filebuf< _CharT, _Traits >::sync ( ) [protected],[virtual]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 893 of file `fstream.tcc`.

4.615.3.41 `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( )`  
`[inline], [protected], [virtual], [inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 700 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sbumpc()`.

4.615.3.42 `template<typename _CharT, typename _Traits> basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ( )`  
`[protected], [virtual]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 204 of file `fstream.tcc`.

References `std::ios_base::in`, and `std::min()`.

4.615.3.43 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (`  
`char_type* __s, streamsize __n )` `[protected], [virtual]`

Multiple character extraction.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 549 of file `fstream.tcc`.

References `std::ios_base::in`.

**4.615.3.44** `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::xsputn ( const char_type * __s, streamsize __n )` `[protected]`, `[virtual]`

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 637 of file `fstream.tcc`.

References `std::min()`, and `std::ios_base::out`.

**4.615.4 Member Data Documentation**

**4.615.4.1** `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf` `[protected]`

Pointer to the beginning of internal buffer.

Definition at line 114 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**4.615.4.2** `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale` `[protected]`, `[inherited]`

Current locale setting.

Definition at line 192 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`, `std::basic_streambuf< char_type, traits_type >::getloc()`, and `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**4.615.4.3** `template<typename _CharT, typename _Traits> size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size`  
[protected]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 121 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**4.615.4.4** `template<typename _CharT, typename _Traits> char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`  
[protected]

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 156 of file fstream.

**4.615.4.5** `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size`  
[protected]

Size of buffer held by `_M_ext_buf`.

Definition at line 161 of file fstream.

**4.615.4.6** `template<typename _CharT, typename _Traits> const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next`  
[protected]

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 168 of file fstream.

**4.615.4.7** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg`  
[protected], [inherited]

Start of get area.

Definition at line 184 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::eback()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**4.615.4.8** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur`  
[protected], [inherited]

Current read area.

Definition at line 185 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::gbump()`, `std::basic_streambuf< char_type, traits_type >::gptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**4.615.4.9** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end`  
[protected], [inherited]

End of get area.

Definition at line 186 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

4.615.4.10 `template<typename _CharT, typename _Traits> ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode` [protected]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 99 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

4.615.4.11 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg` [protected], [inherited]

Start of put area.

Definition at line 187 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbase()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

4.615.4.12 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur` [protected], [inherited]

Current put area.

Definition at line 188 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbump()`, `std::basic_streambuf< char_type, traits_type >::pptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

4.615.4.13 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end` [protected], [inherited]

End of put area.

Definition at line 189 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::epptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

4.615.4.14 `template<typename _CharT, typename _Traits> char_type std::basic_filebuf< _CharT, _Traits >::_M_pback` [protected]

Necessary bits for putback buffer management.

#### Note

pbacks of over one character are not currently supported.

Definition at line 142 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`.

4.615.4.15 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save` [protected]

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 143 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), and std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback().

4.615.4.16 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 144 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), and std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback().

4.615.4.17 `template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init [protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 145 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), and std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback().

4.615.4.18 `template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::_M_reading [protected]`

\_M\_reading == false && \_M\_writing == false for **uncommitted** mode; \_M\_reading == true for **read** mode; \_M\_writing == true for **write** mode;

NB: \_M\_reading == true && \_M\_writing == true is unused.

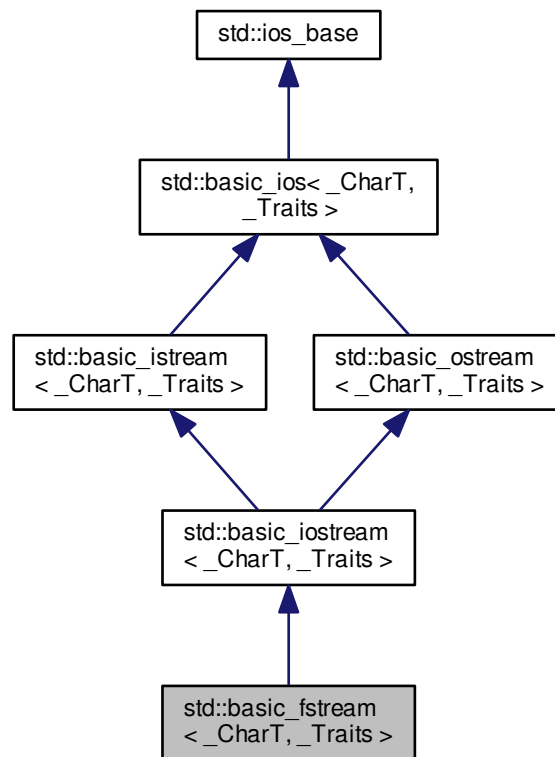
Definition at line 133 of file fstream.

The documentation for this class was generated from the following files:

- [fstream](#)
- [fstream.tcc](#)

## 4.616 std::basic\_fstream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_fstream< \_CharT, \_Traits >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< char_type, traits_type > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`

- typedef num\_put<\_CharT, ostreambuf\_iterator<\_CharT, \_Traits>> \_\_num\_put\_type
- typedef basic\_ostream<\_CharT, \_Traits> \_\_ostream\_type
- typedef basic\_streambuf<\_CharT, \_Traits> \_\_streambuf\_type
- typedef basic\_streambuf<\_CharT, \_Traits> \_\_streambuf\_type
- typedef \_CharT char\_type
- enum event { erase\_event, imbue\_event, copyfmt\_event }
- typedef void(\* event\_callback)(event \_\_e, ios\_base & \_\_b, int \_\_i)
- typedef \_ios\_Fmtflags fmtflags
- typedef traits\_type::int\_type int\_type
- typedef int io\_state
- typedef \_ios\_istate iostate
- typedef traits\_type::off\_type off\_type
- typedef int open\_mode
- typedef \_ios\_Openmode openmode
- typedef traits\_type::pos\_type pos\_type
- typedef int seek\_dir
- typedef \_ios\_Seekdir seekdir
- typedef std::streamoff streamoff
- typedef std::streampos streampos
- typedef \_Traits traits\_type
  
- typedef num\_put<\_CharT, ostreambuf\_iterator<\_CharT, \_Traits>> \_\_num\_put\_type

#### Public Member Functions

- basic\_fstream ()
- basic\_fstream (const char \* \_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- basic\_fstream (const std::string & \_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- ~basic\_fstream ()
- const locale & \_M\_getloc () const
- void \_M\_setstate (iostate \_\_state)
- bool bad () const
- void clear (iostate \_\_state=goodbit)
- void close ()
- basic\_ios & copyfmt (const basic\_ios & \_\_rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate \_\_except)
- bool fail () const
- char\_type fill () const
- char\_type fill (char\_type \_\_ch)
- fmtflags flags () const
- fmtflags flags (fmtflags \_\_fmtfl)
- \_\_ostream\_type & flush ()

- `streamsize gcount ()` const
- `locale getloc ()` const
- `bool good ()` const
- `locale imbue (const locale &__loc)`
- `bool is_open ()`
- `bool is_open ()` const
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __default) const`
- `void open (const char * __s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `void open (const std::string & __s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `__ostream_type & operator<< (const void * __p)`
- `__ostream_type & operator<< (__streambuf_type * __sb)`
- `__istream_type & operator>> (void *& __p)`
- `__istream_type & operator>> (__streambuf_type * __sb)`
- `streamsize precision ()` const
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
- `__filebuf_type * rdbuf ()` const
- `iosstate rdstate ()` const
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `pos_type tellp ()`
- `basic_ostream< _CharT, _Traits > * tie ()` const
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width ()` const
- `streamsize width (streamsize __wide)`
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool & __n)`
- `__istream_type & operator>> (short & __n)`
- `__istream_type & operator>> (unsigned short & __n)`
- `__istream_type & operator>> (int & __n)`

- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type` [get](#) ()
- [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
- [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
- [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ()
- `int_type` [peek](#) ()
- [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
- [\\_\\_istream\\_type](#) & [unget](#) ()
- `int` [sync](#) ()
- `pos_type` [tellg](#) ()
- [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
- [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- [operator void \\*](#) () const
- `bool` [operator!](#) () const
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)

- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

#### Protected Types

- enum { `_S_local_word_size` }

#### Protected Member Functions

- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
[\\_\\_istream\\_type](#) & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT >  
[\\_\\_ostream\\_type](#) & `_M_insert` (\_ValueT \_\_v)
- void `init` ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)

#### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const [\\_\\_ctype\\_type](#) \* `_M_ctype`
- [iostate](#) `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- [fmtflags](#) `_M_flags`
- [streamsize](#) `_M_gcount`
- [locale](#) `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const [\\_\\_num\\_get\\_type](#) \* `_M_num_get`
- const [\\_\\_num\\_put\\_type](#) \* `_M_num_put`
- [streamsize](#) `_M_precision`
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* `_M_streambuf`
- [iostate](#) `_M_streambuf_state`
- [basic\\_ostream](#)<\_CharT, \_Traits> \* `_M_tie`
- [streamsize](#) `_M_width`

- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

#### 4.616.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_fstream< _CharT, _Traits >
```

Controlling input and output for files.

##### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 776 of file `fstream`.

#### 4.616.2 Member Typedef Documentation

4.616.2.1 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]`

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.616.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

##### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.616.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`

- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.616.2.4 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 330 of file `ios_base.h`.

#### 4.616.2.5 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`

- out
- trunc

Definition at line 361 of file ios\_base.h.

#### 4.616.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios\_base.h.

### 4.616.3 Member Enumeration Documentation

#### 4.616.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file ios\_base.h.

### 4.616.4 Constructor & Destructor Documentation

#### 4.616.4.1 template<typename \_CharT, typename \_Traits> std::basic\_fstream<\_CharT, \_Traits>::basic\_fstream ( ) [inline]

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 803 of file fstream.

References `std::basic_ios<_CharT, _Traits>::init()`.

#### 4.616.4.2 template<typename \_CharT, typename \_Traits> std::basic\_fstream<\_CharT, \_Traits>::basic\_fstream ( const char \* \_\_s, ios\_base::openmode \_\_mode = ios\_base::in | ios\_base::out ) [inline], [explicit]

Create an input/output file stream.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 816 of file fstream.

References std::basic\_ios< \_CharT, \_Traits >::init(), and std::basic\_fstream< \_CharT, \_Traits >::open().

**4.616.4.3** `template<typename _CharT, typename _Traits> std::basic_fstream< _CharT, _Traits >::basic_fstream ( const std::string & __s, ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [explicit]`

Create an input/output file stream.

#### Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.  |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

Definition at line 831 of file fstream.

References std::basic\_ios< \_CharT, \_Traits >::init(), and std::basic\_fstream< \_CharT, \_Traits >::open().

**4.616.4.4** `template<typename _CharT, typename _Traits> std::basic_fstream< _CharT, _Traits >::~~basic_fstream ( ) [inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 846 of file fstream.

#### 4.616.5 Member Function Documentation

**4.616.5.1** `const locale& std::ios_base::M_getloc ( ) const [inline], [inherited]`

Locale access.

#### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::time\_put< \_CharT, \_Outlter >::put().

**4.616.5.2** `template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::M_write ( const char_type * __s, streamsize __n ) [inline], [inherited]`

Core write functionality, without sentry.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 311 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

**4.616.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

**4.616.5.4** `template<typename _CharT, typename _Traits > void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.616.5.5** `template<typename _CharT, typename _Traits > void std::basic_fstream< _CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls std::basic\_filebuf::close(). If that function fails, failbit is set in the stream's error state.

Definition at line 926 of file fstream.

References std::basic\_filebuf< \_CharT, \_Traits >::close(), std::ios\_base::failbit, and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.616.5.6** `template<typename _CharT, typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of \_\_rhs into this.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, `std::tie()`, and `std::ios_base::width()`.

**4.616.5.7** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const` `[inline]`, `[inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file `basic_ios.h`.

**4.616.5.8** `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const` `[inline]`, `[inherited]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**4.616.5.9** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except )` `[inline]`, `[inherited]`

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

**4.616.5.10** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const`  
`[inline], [inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::operator void \*(), std::basic\_ios<char, \_Traits>::operator!(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_istream<\_CharT, \_Traits>::tellg(), std::basic\_ostream<\_CharT, \_Traits>::tellp(), and std::regex\_traits<\_Ch\_type>::value().

**4.616.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill ( ) const`  
`[inline], [inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::basic\_ios<char, \_Traits>::fill().

**4.616.5.12** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill ( char_type __ch )`  
`[inline], [inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file basic\_ios.h.

**4.616.5.13** `fmtflags std::ios_base::flags ( ) const` `[inline],[inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.616.5.14** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline],[inherited]`

Setting new format flags all at once.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

#### Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file ios\_base.h.

**4.616.5.15** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::flush ( )` `[inherited]`

Synchronizing the stream buffer.

#### Returns

`*this`

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( )->pubsync ( )`, and if that returns -1, sets `badbit`.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**4.616.5.16** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits>::gcount ( )` `const` `[inline],[inherited]`

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

**4.616.5.17** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get( void ) [inherited]`

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.616.5.18** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get( char_type & __c ) [inherited]`

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.616.5.19** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | Pointer to an array.                          |
| <code>__n</code>     | Maximum number of characters to store in __s. |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.616.5.20** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline], [inherited]`

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s, __n, widen("\n"))`.

Definition at line 354 of file istream.

**4.616.5.21** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::sputc()`.

**4.616.5.22** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline],[inherited]`

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

**4.616.5.23** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.616.5.24** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline], [inherited]`

String extraction.

#### Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

#### Returns

`*this`

Returns `getline(__s, __n, widen('\n'))`.

Definition at line 427 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**4.616.5.25** `locale std::ios_base::getloc ( ) const [inline], [inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::ws()`.

**4.616.5.26** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.616.5.27** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim ) [inherited]`

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.616.5.28** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

4.616.5.29 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( void )` [inherited]

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.616.5.30 `template<typename _CharT, typename _Traits > locale basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )` [inherited]

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls ios\_base::imbue(loc), and if a stream buffer is associated with this stream, calls that buffer's pubimbue(loc).

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file basic\_ios.tcc.

References std::ios\_base::imbue().

Referenced by std::operator<<().

4.616.5.31 `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init ( basic_streambuf< _CharT, _Traits > * __sb )` [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_ios< char, \_Traits >::basic\_ios(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >::basic\_ostreamstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

4.616.5.32 `template<typename _CharT, typename _Traits > bool std::basic_fstream< _CharT, _Traits >::is_open ( )` [inline]

Wrapper to test for an open file.

**Returns**

```
rdbuf()->is_open()
```

Definition at line 865 of file fstream.

References std::basic\_filebuf<\_CharT, \_Traits>::is\_open().

**4.616.5.33 long& std::ios\_base::iword ( int \_\_ix ) [inline], [inherited]**

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios\_base.h.

**4.616.5.34 template<typename \_CharT, typename \_Traits> char std::basic\_ios<\_CharT, \_Traits>::narrow ( char\_type \_\_c, char \_\_default ) const [inline], [inherited]**

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 424 of file basic\_ios.h.

**4.616.5.35 template<typename \_CharT, typename \_Traits> void std::basic\_fstream<\_CharT, \_Traits>::open ( const char \* \_\_s, ios\_base::openmode \_\_mode = ios\_base::in | ios\_base::out ) [inline]**

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 886 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::failbit`, `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`.

**4.616.5.36** `template<typename _CharT, typename _Traits> void std::basic_fstream< _CharT, _Traits >::open ( const std::string & __s, ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline]`

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 907 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::failbit`, `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.616.5.37** `template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::operator void * ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

**4.616.5.38** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 119 of file `basic_ios.h`.

**4.616.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

4.616.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ios_type &(*)(__ios_type &) _pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

4.616.5.41 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) _pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

4.616.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long _n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

4.616.5.43 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long _n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

4.616.5.44 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( bool _n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file `ostream`.

**4.616.5.45** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**4.616.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned short __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**4.616.5.47** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios\_base::basefield, std::ios\_base::flags(), std::ios\_base::hex, and std::ios\_base::oct.

**4.616.5.48** template<typename \_CharT, typename \_Traits> \_\_ostream\_type& std::basic\_ostream< \_CharT, \_Traits >::operator<<( unsigned int \_\_n ) [inline],[inherited]

Integer arithmetic inserters.

**Parameters**

|     |                                      |
|-----|--------------------------------------|
| __n | A variable of builtin integral type. |
|-----|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 192 of file ostream.

**4.616.5.49** template<typename \_CharT, typename \_Traits> \_\_ostream\_type& std::basic\_ostream< \_CharT, \_Traits >::operator<<( long long \_\_n ) [inline],[inherited]

Integer arithmetic inserters.

**Parameters**

|     |                                      |
|-----|--------------------------------------|
| __n | A variable of builtin integral type. |
|-----|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 201 of file ostream.

**4.616.5.50** template<typename \_CharT, typename \_Traits> \_\_ostream\_type& std::basic\_ostream< \_CharT, \_Traits >::operator<<( unsigned long long \_\_n ) [inline],[inherited]

Integer arithmetic inserters.

**Parameters**

|     |                                      |
|-----|--------------------------------------|
| __n | A variable of builtin integral type. |
|-----|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 205 of file ostream.

```
4.616.5.51 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(double __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

```
4.616.5.52 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(float __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

```
4.616.5.53 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(long double __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

```
4.616.5.54 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(const void* __p) [inline],[inherited]
```

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.616.5.55** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< ( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.616.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline],[inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

**4.616.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline],[inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

4.616.5.58 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( ios_base &(*)(ios_base &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

4.616.5.59 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( bool & __n ) [inline], [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

4.616.5.60 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::operator>>( short & __n ) [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.616.5.61 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( unsigned short & __n ) [inline], [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.616.5.62** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::operator>> ( int & __n ) [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.616.5.63** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.616.5.64** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 186 of file istream.

```
4.616.5.65 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> (unsigned long & __n) [inline],[inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 190 of file istream.

```
4.616.5.66 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> (long long & __n) [inline],[inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 195 of file istream.

```
4.616.5.67 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> (unsigned long long & __n) [inline],[inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 199 of file istream.

4.616.5.68 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( float & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

4.616.5.69 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( double & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

4.616.5.70 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( long double & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

4.616.5.71 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( void *& __p ) [inline],[inherited]`

Basic arithmetic extractors.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.616.5.72** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, `failbit` is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

**4.616.5.73** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::int_type basic_istream< _CharT, _Traits>::peek ( void ) [inherited]`

Looking ahead in the stream.

## Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.616.5.74 **streamsize** std::ios\_base::precision ( ) const [inline],[inherited]

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::operator<<().

4.616.5.75 **streamsize** std::ios\_base::precision ( streamsize \_\_prec ) [inline],[inherited]

Changing flags.

#### Parameters

|               |                          |
|---------------|--------------------------|
| <b>__prec</b> | The new precision value. |
|---------------|--------------------------|

#### Returns

The previous value of precision().

Definition at line 630 of file ios\_base.h.

4.616.5.76 **template**<typename \_CharT, typename \_Traits> **basic\_ostream**<\_CharT, \_Traits> & **basic\_ostream**<\_CharT, \_Traits>::put ( char\_type \_\_c ) [inherited]

Simple insertion.

#### Parameters

|            |                          |
|------------|--------------------------|
| <b>__c</b> | The character to insert. |
|------------|--------------------------|

#### Returns

\*this

Tries to insert \_\_c.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), and std::basic\_ios<\_CharT, \_Traits>::setstate().

Referenced by std::endl(), and std::ends().

4.616.5.77 **template**<typename \_CharT, typename \_Traits> **basic\_istream**<\_CharT, \_Traits> & **basic\_istream**<\_CharT, \_Traits>::putback ( char\_type \_\_c ) [inherited]

Unextracting a single character.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

## Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

## Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**4.616.5.78** `void*& std::ios_base::pword ( int __ix )` `[inline]`, `[inherited]`

Access to void pointer array.

## Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

## Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file `ios_base.h`.

**4.616.5.79** `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf ( basic_streambuf< _CharT, _Traits > * __sb )` `[inherited]`

Changing the underlying buffer.

## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.616.5.80** `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_fstream<_CharT, _Traits>::rdbuf ( ) const` `[inline]`

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 857 of file `fstream`.

**4.616.5.81** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const` `[inline]`, `[inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**4.616.5.82** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::read ( char_type * __s, streamsize __n )` `[inherited]`

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.616.5.83** `template<typename _CharT, typename _Traits> streamsize basic_istream< _CharT, _Traits >::readsome (char_type * __s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.616.5.84** `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.616.5.85 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.616.5.86 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.616.5.87 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::seekp ( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.616.5.88 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.616.5.89 `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline], [inherited]`

Setting new format flags.

#### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

4.616.5.90 **fmtflags** std::ios\_base::setf ( *fmtflags* \_\_*fmtfl*, *fmtflags* \_\_*mask* ) [inline], [inherited]

Setting new format flags.

#### Parameters

|                 |                                   |
|-----------------|-----------------------------------|
| __ <i>fmtfl</i> | Additional flags to set.          |
| __ <i>mask</i>  | The flags mask for <i>fmtfl</i> . |

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

4.616.5.91 **template<typename \_CharT, typename \_Traits> void** std::basic\_ios< \_CharT, \_Traits >::setstate ( *iostate* \_\_*state* ) [inline], [inherited]

Sets additional flags in the error state.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| __ <i>state</i> | The additional state flag(s) to set. |
|-----------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file `basic_ios.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::tr2::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

4.616.5.92 **template<typename \_CharT, typename \_Traits> int** basic\_istream< \_CharT, \_Traits >::sync ( *void* ) [inherited]

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.616.5.93** `static bool std::ios_base::sync_with_stdio( bool __sync = true )` `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.616.5.94** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg( void )` `[inherited]`

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 817 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**4.616.5.95** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp( )` `[inherited]`

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::cur, std::basic\_ios<\_CharT, \_Traits>::fail(), std::ios\_base::out, and std::basic\_ios<\_CharT, \_Traits>::rdbuf().

4.616.5.96 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline], [inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::basic\_ostream<\_CharT, \_Traits>::sentry::sentry(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

4.616.5.97 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( basic_ostream<_CharT, _Traits>* __tiestr ) [inline], [inherited]`

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic\_ios.h.

4.616.5.98 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::unget ( void ) [inherited]`

Unextracting the previous character.

#### Returns

\*this

If rdbuf() is not null, calls rdbuf()->sungetc().

If rdbuf() is null or if sungetc() fails, sets badbit in the error state.

#### Note

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::clear(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::rdstate(), std::basic\_ios< \_CharT, \_Traits >::setstate(), and std::basic\_streambuf< \_CharT, \_Traits >::sungetc().

**4.616.5.99** void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline],[inherited]

Clearing format flags.

#### Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**4.616.5.100** template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios< \_CharT, \_Traits >::widen ( char \_\_c ) const [inline],[inherited]

Widens characters.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios< char, \_Traits >::fill(), std::basic\_istream< char >::get(), std::basic\_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

**4.616.5.101** streamsize std::ios\_base::width ( ) const [inline],[inherited]

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_Outiter >::do\_put(), and std::operator>>().

4.616.5.102 **streamsize** std::ios\_base::width ( **streamsize** \_\_wide ) [inline],[inherited]

Changing flags.

#### Parameters

|        |                      |
|--------|----------------------|
| __wide | The new width value. |
|--------|----------------------|

#### Returns

The previous value of width().

Definition at line 653 of file ios\_base.h.

4.616.5.103 **template**<typename \_CharT, typename \_Traits> **basic\_ostream**<\_CharT, \_Traits> & **basic\_ostream**<\_CharT, \_Traits>::write ( **const char\_type** \* \_\_s, **streamsize** \_\_n ) [inherited]

Character string insertion.

#### Parameters

|     |                                         |
|-----|-----------------------------------------|
| __s | The array to insert.                    |
| __n | Maximum number of characters to insert. |

#### Returns

\*this

Characters are copied from \_\_s and inserted into the stream until one of the following happens:

- \_\_n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic\_ostream<\_CharT, \_Traits>::\_M\_write(), and std::ios\_base::badbit.

4.616.5.104 **static int** std::ios\_base::xalloc ( ) throw () [static],[inherited]

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

## 4.616.6 Member Data Documentation

**4.616.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount`  
`[protected], [inherited]`

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

**4.616.6.2** `const fmtflags std::ios_base::adjustfield` `[static], [inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.616.6.3** `const openmode std::ios_base::app` `[static], [inherited]`

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.616.6.4** `const openmode std::ios_base::ate` `[static], [inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**4.616.6.5** `const iostate std::ios_base::badbit` `[static], [inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

**4.616.6.6** `const fmtflags std::ios_base::basefield` `[static], [inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<`

`_CharT, _Traits >::operator<<()`.

#### 4.616.6.7 `const seekdir std::ios_base::beg` [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

#### 4.616.6.8 `const openmode std::ios_base::binary` [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

#### 4.616.6.9 `const fmtflags std::ios_base::boolalpha` [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

#### 4.616.6.10 `const seekdir std::ios_base::cur` [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

#### 4.616.6.11 `const fmtflags std::ios_base::dec` [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.616.6.12 `const seekdir std::ios_base::end` [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.616.6.13 `const iostate std::ios_base::eofbit` [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time-`

\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 4.616.6.14 const iostate std::ios\_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.616.6.15 const fmtflags std::ios\_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.616.6.16 const fmtflags std::ios\_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.616.6.17 const iostate std::ios\_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.616.6.18 const fmtflags std::ios\_base::hex** [static],[inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.616.6.19 const openmode std::ios\_base::in** [static],[inherited]

Open for input. Default for ifstream and fstream.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**4.616.6.20 const fmtflags std::ios\_base::internal** [static],[inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

**4.616.6.21 const fmtflags std::ios\_base::left** [static],[inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.616.6.22 const fmtflags std::ios\_base::oct** [static],[inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.616.6.23 const openmode std::ios\_base::out** [static],[inherited]

Open for output. Default for ofstream and fstream.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**4.616.6.24 const fmtflags std::ios\_base::right** [static],[inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios\_base.h.

Referenced by std::right().

**4.616.6.25** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios\_base.h.

Referenced by std::scientific().

**4.616.6.26** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

**4.616.6.27** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**4.616.6.28** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**4.616.6.29** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream<\_CharT, \_Traits>::sentry::sentry(), and std::skipws().

**4.616.6.30** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for ofstream.

Definition at line 381 of file ios\_base.h.

**4.616.6.31** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios\_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic\_ostream<\_CharT, \_Traits>::sentry::~sentry().

**4.616.6.32** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios\_base.h.

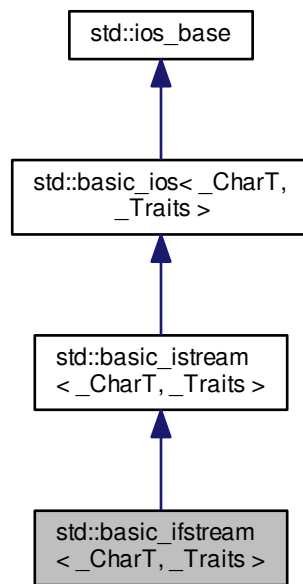
Referenced by std::num\_put<\_CharT, \_Outlter>::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- [fstream](#)

#### 4.617 std::basic\_ifstream< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_ifstream< \_CharT, \_Traits >:



#### Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_filebuf](#)  
< char\_type, traits\_type > **\_\_filebuf\_type**
- typedef [basic\\_ios](#)< \_CharT,  
\_Traits > **\_\_ios\_type**
- typedef [basic\\_istream](#)  
< char\_type, traits\_type > **\_\_istream\_type**
- typedef [num\\_get](#)< \_CharT,  
[istreambuf\\_iterator](#)< \_CharT,  
\_Traits > > **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)  
< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
- typedef void(\* [event\\_callback](#) )(event \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)

- typedef \_ios\_Fmtflags [fmtflags](#)
- typedef traits\_type::int\_type **int\_type**
- typedef int **io\_state**
- typedef \_ios\_istate [istate](#)
- typedef traits\_type::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_ios\_Openmode [openmode](#)
- typedef traits\_type::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_ios\_Seekdir [seekdir](#)
- typedef [std::streamoff](#) **streamoff**
- typedef [std::streampos](#) **streampos**
- typedef \_Traits **traits\_type**
- typedef [num\\_put](#)<\_CharT,  
[ostreambuf\\_iterator](#)<\_CharT,  
\_Traits>> [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_ifstream](#) ()
- [basic\\_ifstream](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [~basic\\_ifstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- bool [is\\_open](#) ()
- bool [is\\_open](#) () const
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- void [open](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- void [open](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [\\_\\_istream\\_type](#) & [operator>>](#) (void \*&\_\_p)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)

- [streamsize precision](#) () const
- [streamsize precision](#) (streamsize \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- [\\_\\_filebuf\\_type](#) \* [rdbuf](#) () const
- [iostate rdstate](#) () const
- void [register\\_callback](#) (event\_callback \_\_fn, int \_\_index)
- [fmtflags setf](#) (fmtflags \_\_fmtfl)
- [fmtflags setf](#) (fmtflags \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) (iostate \_\_state)
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) () const
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) ([basic\\_ostream](#)<\_CharT, \_Traits> \* \_\_tiestr)
- void [unsetf](#) (fmtflags \_\_mask)
- char\_type [widen](#) (char \_\_c) const
- [streamsize width](#) () const
- [streamsize width](#) (streamsize \_\_wide)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_istream\\_type](#) &(\*\_\_pf)(\_\_istream\_type &))
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)(\_\_ios\_type &))
- [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_istream\\_type](#) & [operator>>](#) (bool &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- int\_type [get](#) ()
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ()
  - int\_type [peek](#) ()
  - [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator void \\*](#) () const
  - bool [operator!](#) () const

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)

- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

#### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

#### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT >  
[\\_istream\\_type](#) & [\\_M\\_extract](#) (\_ValueT &\_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [init](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)

#### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

## 4.617.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ifstream< _CharT, _Traits >
```

Controlling input for files.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 427 of file `fstream`.

## 4.617.2 Member Typedef Documentation

4.617.2.1 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]`

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.617.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.617.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`

- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file `ios_base.h`.

#### 4.617.2.4 `typedef _ios_istate std::ios_base::istate` [inherited]

This is a bitmask type.

`_Ios_Istate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `istate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file `ios_base.h`.

#### 4.617.2.5 `typedef _ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file `ios_base.h`.

## 4.617.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

## 4.617.3 Member Enumeration Documentation

## 4.617.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

## 4.617.4 Constructor &amp; Destructor Documentation

## 4.617.4.1 template&lt;typename \_CharT, typename \_Traits&gt; std::basic\_ifstream&lt;\_CharT, \_Traits&gt;::basic\_ifstream ( ) [inline]

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 453 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

## 4.617.4.2 template&lt;typename \_CharT, typename \_Traits&gt; std::basic\_ifstream&lt;\_CharT, \_Traits&gt;::basic\_ifstream ( const char \* \_\_s, ios\_base::openmode \_\_mode = ios\_base::in ) [inline], [explicit]

Create an input file stream.

## Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::in` is automatically included in `__mode`.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 467 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`, and `std::basic_ifstream<_CharT, _Traits>::open()`.

4.617.4.3 `template<typename _CharT, typename _Traits> std::basic_ifstream<_CharT, _Traits>::basic_ifstream( const std::string & __s, ios_base::openmode __mode = ios_base::in ) [inline], [explicit]`

Create an input file stream.

#### Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | std::string specifying the filename.             |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

`ios_base::in` is automatically included in `__mode`.

Definition at line 483 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`, and `std::basic_ifstream<_CharT, _Traits>::open()`.

4.617.4.4 `template<typename _CharT, typename _Traits> std::basic_ifstream<_CharT, _Traits>::~~basic_ifstream( ) [inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 498 of file `fstream`.

#### 4.617.5 Member Function Documentation

4.617.5.1 `const locale& std::ios_base::M_getloc( ) const [inline], [inherited]`

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get_date()`, `std::time_get<_CharT, _Inlter>::do_get_monthname()`, `std::time_get<_CharT, _Inlter>::do_get_time()`, `std::time_get<_CharT, _Inlter>::do_get_weekday()`, `std::time_get<_CharT, _Inlter>::do_get_year()`, `std::time_put<_CharT, _Outlter>::do_put()`, `std::num_put<_CharT, _Outlter>::do_put()`, and `std::time_put<_CharT, _Outlter>::put()`.

4.617.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad( ) const [inline], [inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file `basic_ios.h`.

**4.617.5.3** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**4.617.5.4** `template<typename _CharT, typename _Traits> void std::basic_ifstream< _CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 576 of file `fstream`.

References `std::basic_filebuf< _CharT, _Traits >::close()`, `std::ios_base::failbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.617.5.5** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, `std::tie()`, and `std::ios_base::width()`.

**4.617.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic\_ios.h.

**4.617.5.7** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const`  
`[inline], [inherited]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt().

**4.617.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except`  
`) [inline], [inherited]`

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

**4.617.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const`  
`[inline], [inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_ifstream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ifstream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**4.617.5.10** template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios< \_CharT, \_Traits >::fill ( ) const  
[inline], [inherited]

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 364 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::basic\_ios< char, \_Traits >::fill().

**4.617.5.11** template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios< \_CharT, \_Traits >::fill ( char\_type \_\_ch )  
[inline], [inherited]

Sets a new *empty* character.

**Parameters**

|             |                    |
|-------------|--------------------|
| <u>__ch</u> | The new character. |
|-------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file basic\_ios.h.

**4.617.5.12** fmtflags std::ios\_base::flags ( ) const [inline], [inherited]

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic\_ifstream< \_CharT, \_Traits >::sentry::sentry().

4.617.5.13 **fmtflags** std::ios\_base::flags ( **fmtflags** \_\_fmtfl ) [inline], [inherited]

Setting new format flags all at once.

#### Parameters

|         |                       |
|---------|-----------------------|
| __fmtfl | The new flags to set. |
|---------|-----------------------|

#### Returns

The previous format control flags.

This function overwrites all the format flags with \_\_fmtfl.

Definition at line 562 of file ios\_base.h.

4.617.5.14 **template**<typename \_CharT, typename \_Traits> **streamsize** std::basic\_istream<\_CharT, \_Traits>::gcount ( )  
**const** [inline], [inherited]

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.617.5.15 **template**<typename \_CharT, typename \_Traits> **basic\_istream**<\_CharT, \_Traits>::int\_type **basic\_istream**<\_CharT, \_Traits>::get ( **void** ) [inherited]

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits>::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), and std::basic\_ios<\_CharT, \_Traits>::setstate().

4.617.5.16 **template**<typename \_CharT, typename \_Traits> **basic\_istream**<\_CharT, \_Traits> & **basic\_istream**<\_CharT, \_Traits>::get ( **char\_type** & \_\_c ) [inherited]

Simple extraction.

#### Parameters

|     |                                       |
|-----|---------------------------------------|
| __c | The character in which to store data. |
|-----|---------------------------------------|

#### Returns

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.617.5.17** template<typename \_CharT, typename \_Traits> basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::get ( char\_type \* \_\_s, streamsize \_\_n, char\_type \_\_delim ) [inherited]

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.617.5.18** template<typename \_CharT, typename \_Traits> \_\_istream\_type& std::basic\_istream< \_CharT, \_Traits >::get ( char\_type \* \_\_s, streamsize \_\_n ) [inline], [inherited]

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen('\n'))`.

Definition at line 354 of file istream.

4.617.5.19 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::setstate()`, `std::basic_streambuf< _CharT, _Traits>::sgetc()`, `std::basic_streambuf< _CharT, _Traits>::snextc()`, and `std::basic_streambuf< _CharT, _Traits>::sputc()`.

4.617.5.20 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::get ( __streambuf_type & __sb ) [inline],[inherited]`

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen('\n'))`.

Definition at line 387 of file istream.

4.617.5.21 `template<typename _CharT, typename _Traits> basic_ifstream< _CharT, _Traits> & basic_ifstream< _CharT, _Traits>::getline( char_type* __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

#### Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

#### Returns

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_ifstream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_streambuf< _CharT, _Traits>::sbumpc()`, `std::basic_ios< _CharT, _Traits>::setstate()`, `std::basic_streambuf< _CharT, _Traits>::sgetc()`, and `std::basic_streambuf< _CharT, _Traits>::snextc()`.

4.617.5.22 `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream< _CharT, _Traits>::getline( char_type* __s, streamsize __n ) [inline], [inherited]`

String extraction.

#### Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

#### Returns

`*this`

Returns `getline(__s, __n, widen("\n"))`.

Definition at line 427 of file istream.

Referenced by `std::basic_ifstream< char>::getline()`.

**4.617.5.23** locale std::ios\_base::getloc ( ) const [inline], [inherited]

Locale access.

**Returns**

A copy of the current locale.

If imbue(loc) has previously been called, then this function returns loc. Otherwise, it returns a copy of std::locale(), the global C++ locale.

Definition at line 695 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::money\_put<\_CharT, \_Outiter>::do\_put(), std::operator>>(), and std::ws().

**4.617.5.24** template<typename \_CharT, typename \_Traits> bool std::basic\_ios<\_CharT, \_Traits>::good ( ) const [inline], [inherited]

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic\_ios.h.

Referenced by std::basic\_ostream<\_CharT, \_Traits>::sentry::sentry(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

**4.617.5.25** template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & basic\_istream<\_CharT, \_Traits>::ignore ( streamsize \_\_n, int\_type \_\_delim ) [inherited]

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic\_ifstream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.617.5.26** template<typename \_CharT, typename \_Traits> basic\_ifstream< \_CharT, \_Traits> & basic\_ifstream< \_CharT, \_Traits>::ignore( streamsize \_\_n ) [inherited]

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic\_ifstream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.617.5.27** template<typename \_CharT, typename \_Traits> basic\_ifstream< \_CharT, \_Traits> & basic\_ifstream< \_CharT, \_Traits>::ignore( void ) [inherited]

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic\_ifstream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.617.5.28** template<typename \_CharT, typename \_Traits> locale basic\_ios< \_CharT, \_Traits>::imbue( const locale & \_\_loc ) [inherited]

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls ios\_base::imbue(loc), and if a stream buffer is associated with this stream, calls that buffer's pubimbue(loc).

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file basic\_ios.tcc.

References std::ios\_base::imbue().

Referenced by std::operator<<().

**4.617.5.29** template<typename \_CharT, typename \_Traits> void basic\_ios< \_CharT, \_Traits >::init ( basic\_streambuf< \_CharT, \_Traits > \* \_\_sb ) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_ios< char, \_Traits >::basic\_ios(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >::basic\_ostreamstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

**4.617.5.30** template<typename \_CharT, typename \_Traits > bool std::basic\_ifstream< \_CharT, \_Traits >::is\_open ( ) [inline]

Wrapper to test for an open file.

Returns

rdbuf() -> is\_open()

Definition at line 517 of file fstream.

References std::basic\_filebuf< \_CharT, \_Traits >::is\_open().

**4.617.5.31** long& std::ios\_base::iword ( int \_\_ix ) [inline], [inherited]

Access to integer array.

Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios\_base.h.

**4.617.5.32** template<typename \_CharT, typename \_Traits> char std::basic\_ios< \_CharT, \_Traits >::narrow ( char\_type \_\_c, char \_\_dfault ) const [inline], [inherited]

Squeezes characters.

Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file `basic_ios.h`.

**4.617.5.33** `template<typename _CharT, typename _Traits> void std::basic_ifstream<_CharT, _Traits>::open ( const char * __s, ios_base::openmode __mode = ios_base::in ) [inline]`

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s, __mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 538 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::ios_base::in`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`.

**4.617.5.34** `template<typename _CharT, typename _Traits> void std::basic_ifstream<_CharT, _Traits>::open ( const std::string & __s, ios_base::openmode __mode = ios_base::in ) [inline]`

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 558 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::ios_base::in`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.617.5.35** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

```
4.617.5.36 template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! () const
[inline],[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 119 of file `basic_ios.h`.

```
4.617.5.37 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__istream_type &(*)(__istream_type &) __pf) [inline],[inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

```
4.617.5.38 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline],[inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

```
4.617.5.39 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (ios_base &(*)(ios_base &) __pf) [inline],[inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file `istream`.

```
4.617.5.40 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (bool & __n) [inline],[inherited]
```

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

**4.617.5.41** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & basic_ifstream<_CharT, _Traits>::operator>> ( short & __n ) [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.617.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> ( unsigned short & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.617.5.43** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & basic_ifstream<_CharT, _Traits>::operator>> ( int & __n ) [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.617.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.617.5.45** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**4.617.5.46** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 190 of file istream.

**4.617.5.47** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 195 of file istream.

**4.617.5.48** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 199 of file istream.

**4.617.5.49** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( float & __f ) [inline], [inherited]`

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 214 of file istream.

4.617.5.50 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( double & __f ) [inline], [inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

4.617.5.51 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( long double & __f ) [inline], [inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

4.617.5.52 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( void *& __p ) [inline], [inherited]`

Basic arithmetic extractors.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

4.617.5.53 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::operator>>( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.617.5.54** `template<typename _CharT, typename _Traits> basic_ifstream< _CharT, _Traits >::int_type basic_ifstream< _CharT, _Traits >::peek ( void ) [inherited]`

Looking ahead in the stream.

## Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_ifstream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.617.5.55** `streamsize std::ios_base::precision ( ) const [inline],[inherited]`

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**4.617.5.56** `streamsize std::ios_base::precision ( streamsize __prec ) [inline],[inherited]`

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

**4.617.5.57** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & basic_ifstream<_CharT, _Traits>::putback ( char_type __c ) [inherited]`

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_ifstream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

**4.617.5.58** `void*& std::ios_base::pword ( int __ix ) [inline], [inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

**4.617.5.59** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.617.5.60** `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_ifstream<_CharT, _Traits>::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 509 of file fstream.

**4.617.5.61** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline], [inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file basic\_ios.h.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

4.617.5.62 `template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & basic_ifstream< _CharT, _Traits >::read ( char_type * __s, streamsize __n ) [inherited]`

Extraction without delimiters.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

#### Returns

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_ifstream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.617.5.63 `template<typename _CharT, typename _Traits > streamsize basic_ifstream< _CharT, _Traits >::readsome ( char_type * __s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

#### Returns

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the `streambuf`'s buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the `streambuf`.

Definition at line 679 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::min(), std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.617.5.64 void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]

Add the callback \_\_fn with parameter \_\_index.

#### Parameters

|         |                                                   |
|---------|---------------------------------------------------|
| __fn    | The function to add.                              |
| __index | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.617.5.65 template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::seekg ( pos\_type \_\_pos ) [inherited]

Changing the current read position.

#### Parameters

|       |                         |
|-------|-------------------------|
| __pos | A file position object. |
|-------|-------------------------|

#### Returns

\*this

If fail() is not true, calls rdbuf()->pubseekpos(\_\_pos). If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount().

Definition at line 845 of file istream.tcc.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::clear(), std::ios\_base::eofbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::in, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::rdstate(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.617.5.66 template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::seekg ( off\_type \_\_off, ios\_base::seekdir \_\_dir ) [inherited]

Changing the current read position.

#### Parameters

|       |                                 |
|-------|---------------------------------|
| __off | A file offset object.           |
| __dir | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.617.5.67** `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline]`, `[inherited]`

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.617.5.68** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

**Parameters**

|                      |                                           |
|----------------------|-------------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                  |
| <code>__mask</code>  | The flags mask for <code>__fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `__fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.617.5.69** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )` `[inline]`, `[inherited]`

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file `basic_ios.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::tr2::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**4.617.5.70** `template<typename _CharT, typename _Traits> int basic_istream< _CharT, _Traits >::sync ( void )`  
[inherited]

Synchronizing the stream buffer.

## Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

## Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.617.5.71** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )` [static], [inherited]

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.617.5.72** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits>::pos_type basic_ifstream<_CharT, _Traits>::tellg( void )` [inherited]

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**4.617.5.73** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( ) const` [inline], [inherited]

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_ifstream<_CharT, _Traits>::sentry::sentry()`.

**4.617.5.74** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( basic_ostream<_CharT, _Traits>* __tiestr )` [inline], [inherited]

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic\_ios.h.

**4.617.5.75** `template<typename _CharT, typename _Traits> basic_ifstream< _CharT, _Traits> & basic_ifstream< _CharT, _Traits>::unget ( void ) [inherited]`

Unextracting the previous character.

**Returns**

\*this

If rdbuf() is not null, calls rdbuf()->sungetc(c).

If rdbuf() is null or if sungetc() fails, sets badbit in the error state.

**Note**

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic\_ifstream< \_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits>::clear(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits>::rdbuf(), std::basic\_ios< \_CharT, \_Traits>::rdstate(), std::basic\_ios< \_CharT, \_Traits>::setstate(), and std::basic\_streambuf< \_CharT, \_Traits>::sungetc().

**4.617.5.76** `void std::ios_base::unsetf ( fmtflags __mask ) [inline], [inherited]`

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**4.617.5.77** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits>::widen ( char __c ) const [inline], [inherited]`

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios<char, _Traits>::fill()`, `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::getline()`, `std::tr2::operator>>()`, and `std::operator>>()`.

**4.617.5.78** `streamsize std::ios_base::width ( ) const` `[inline]`, `[inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**4.617.5.79** `streamsize std::ios_base::width ( streamsize __wide )` `[inline]`, `[inherited]`

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

**4.617.5.80** `static int std::ios_base::xalloc ( ) throw ()` `[static]`, `[inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iwor`d and `pwor`d functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iwor`d and `pwor`d arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

#### 4.617.6 Member Data Documentation

**4.617.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount`  
[protected], [inherited]

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

**4.617.6.2** `const fmtflags std::ios_base::adjustfield` [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.617.6.3** `const openmode std::ios_base::app` [static], [inherited]

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.617.6.4** `const openmode std::ios_base::ate` [static], [inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**4.617.6.5** `const iostate std::ios_base::badbit` [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

**4.617.6.6** `const fmtflags std::ios_base::basefield` [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

#### 4.617.6.7 `const seekdir std::ios_base::beg` `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

#### 4.617.6.8 `const openmode std::ios_base::binary` `[static]`, `[inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

#### 4.617.6.9 `const fmtflags std::ios_base::boolalpha` `[static]`, `[inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

#### 4.617.6.10 `const seekdir std::ios_base::cur` `[static]`, `[inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

#### 4.617.6.11 `const fmtflags std::ios_base::dec` `[static]`, `[inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.617.6.12 `const seekdir std::ios_base::end` `[static]`, `[inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.617.6.13 `const iostate std::ios_base::eofbit` `[static]`, `[inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsomewhat(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 4.617.6.14 const iostate std::ios\_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.617.6.15 const fmtflags std::ios\_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.617.6.16 const fmtflags std::ios\_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.617.6.17 const iostate std::ios\_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsomewhat(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.617.6.18 const fmtflags std::ios\_base::hex** [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.617.6.19 const openmode std::ios\_base::in** [static], [inherited]

Open for input. Default for ifstream and fstream.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**4.617.6.20 const fmtflags std::ios\_base::internal** [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

**4.617.6.21 const fmtflags std::ios\_base::left** [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.617.6.22 const fmtflags std::ios\_base::oct** [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.617.6.23 const openmode std::ios\_base::out** [static], [inherited]

Open for output. Default for ofstream and fstream.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**4.617.6.24 const fmtflags std::ios\_base::right** [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.617.6.25** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.617.6.26** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.617.6.27** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.617.6.28** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.617.6.29** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_ifstream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

**4.617.6.30** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.617.6.31** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

**4.617.6.32** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

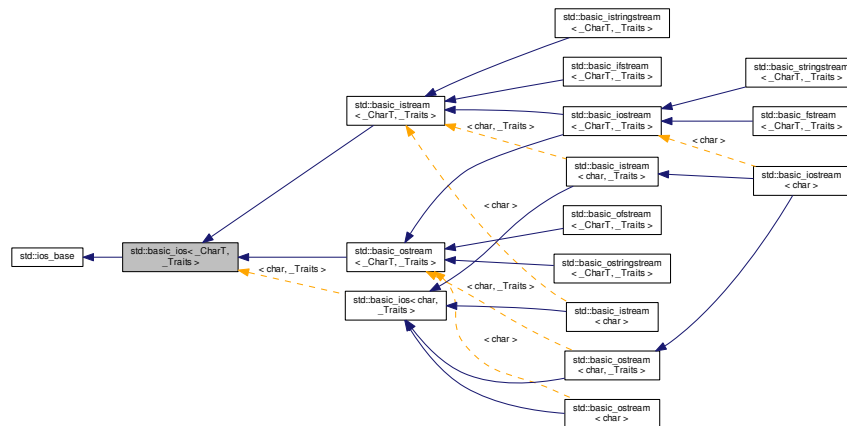
Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [fstream](#)

## 4.618 std::basic\_ios< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_ios< \_CharT, \_Traits >:



### Public Types

- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#))([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
- typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)
- typedef int [io\\_state](#)
- typedef [\\_ios\\_istate](#) [iostate](#)
- typedef int [open\\_mode](#)
- typedef [\\_ios\\_Openmode](#) [openmode](#)
- typedef int [seek\\_dir](#)
- typedef [\\_ios\\_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)

- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits::int\\_type](#) [int\\_type](#)
- typedef [\\_Traits::pos\\_type](#) [pos\\_type](#)
- typedef [\\_Traits::off\\_type](#) [off\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)

- typedef [ctype](#)< [\\_CharT](#) > [\\_\\_ctype\\_type](#)
- typedef [num\\_put](#)< [\\_CharT](#), [ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_put\\_type](#)
- typedef [num\\_get](#)< [\\_CharT](#), [istreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_get\\_type](#)

## Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
  - virtual [~basic\\_ios](#) ()
  - const [locale](#) & [\\_M\\_getloc](#) () const
  - void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
  - bool [bad](#) () const
  - void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
  - [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
  - bool [eof](#) () const
  - [iostate](#) [exceptions](#) () const
  - void [exceptions](#) ([iostate](#) \_\_except)
  - bool [fail](#) () const
  - [char\\_type](#) [fill](#) () const
  - [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
  - [fmtflags](#) [flags](#) () const
  - [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
  - [locale](#) [getloc](#) () const
  - bool [good](#) () const
  - [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
  - long & [iword](#) (int \_\_ix)
  - [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) () const
  - [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
  - [iostate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type](#) [widen](#) ([char](#) \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [operator void \\*](#) () const
  - bool [operator!](#) () const

## Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- `basic_ios` ()
- void `_M_cache_locale` (const `locale` & \_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [ _S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 4.618.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ios< _CharT, _Traits >
```

Template class `basic_ios`, virtual base class for all stream classes.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 66 of file `basic_ios.h`.

## 4.618.2 Member Typedef Documentation

4.618.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type`

These are non-standard types.

Definition at line 86 of file `basic_ios.h`.

4.618.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type`

These are non-standard types.

Definition at line 90 of file `basic_ios.h`.

4.618.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ios< _CharT, _Traits >::__num_put_type`

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.618.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ios< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 75 of file `basic_ios.h`.

4.618.2.5 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.618.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

**4.618.2.7** `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 76 of file `basic_ios.h`.

**4.618.2.8** `typedef _Ios_Iostate std::ios_base::iostate` `[inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 330 of file `ios_base.h`.

**4.618.2.9** `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ios<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 78 of file `basic_ios.h`.

**4.618.2.10** `typedef _Ios_Openmode std::ios_base::openmode` `[inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 361 of file `ios_base.h`.

**4.618.2.11** `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 77 of file `basic_ios.h`.

**4.618.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir` `[inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

**4.618.2.13** `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 79 of file `basic_ios.h`.

### 4.618.3 Member Enumeration Documentation

**4.618.3.1** `enum std::ios_base::event` `[inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

### 4.618.4 Constructor & Destructor Documentation

**4.618.4.1** `template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::basic_ios ( basic_streambuf< _CharT, _Traits > * _sb )` `[inline], [explicit]`

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 264 of file `basic_ios.h`.

**4.618.4.2** `template<typename _CharT, typename _Traits> virtual std::basic_ios< _CharT, _Traits >::~basic_ios ( )` `[inline], [virtual]`

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by `rdbuf()`.

Definition at line 276 of file `basic_ios.h`.

4.618.4.3 `template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::basic_ios ( ) [inline], [protected]`

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 454 of file `basic_ios.h`.

#### 4.618.5 Member Function Documentation

4.618.5.1 `const locale& std::ios_base::M.getloc ( ) const [inline], [inherited]`

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

4.618.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline]`

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file `basic_ios.h`.

4.618.5.3 `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )`

[Re]sets the error state.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

4.618.5.4 `template<typename _CharT, typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs )`

Copies fields of `__rhs` into this.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, `std::tie()`, and `std::ios_base::width()`.

4.618.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 184 of file `basic_ios.h`.

4.618.5.6 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

4.618.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except ) [inline]`

Throwing exceptions on errors.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file `basic_ios.h`.

**4.618.5.8** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const` `[inline]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator void *()`, `std::basic_ios< char, _Traits >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

**4.618.5.9** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const` `[inline]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

**4.618.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )` `[inline]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file `basic_ios.h`.

**4.618.5.11** `fmtflags std::ios_base::flags ( ) const` `[inline],[inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.618.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline],[inherited]`

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

**4.618.5.13** `locale std::ios_base::getloc ( ) const` `[inline],[inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::operator>>()`, and `std::ws()`.

4.618.5.14 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const`  
`[inline]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.618.5.15 `template<typename _CharT, typename _Traits > locale basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )`

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file `basic_ios.tcc`.

References `std::ios_base::imbue()`.

Referenced by `std::operator<<()`.

4.618.5.16 `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init ( basic_streambuf< _CharT, _Traits > * __sb )` `[protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.

4.618.5.17 `long& std::ios_base::iword ( int __ix )` `[inline]`, `[inherited]`

Access to integer array.

## Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

## Returns

A reference to an integer associated with the index.

The `ixword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

**4.618.5.18** `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char __dfault ) const [inline]`

Squeezes characters.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file `basic_ios.h`.

**4.618.5.19** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**4.618.5.20** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file `basic_ios.h`.

4.618.5.21 **streamsize** std::ios\_base::precision ( ) const [inline],[inherited]

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::operator<<().

4.618.5.22 **streamsize** std::ios\_base::precision ( **streamsize** \_\_prec ) [inline],[inherited]

Changing flags.

#### Parameters

|        |                          |
|--------|--------------------------|
| __prec | The new precision value. |
|--------|--------------------------|

#### Returns

The previous value of precision().

Definition at line 630 of file ios\_base.h.

4.618.5.23 **void\*&** std::ios\_base::pword ( int \_\_ix ) [inline],[inherited]

Access to void pointer array.

#### Parameters

|      |                       |
|------|-----------------------|
| __ix | Index into the array. |
|------|-----------------------|

#### Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

4.618.5.24 **template<typename** \_CharT, **typename** \_Traits> **basic\_streambuf<\_CharT, \_Traits>\*** std::basic\_ios< \_CharT, \_Traits >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.618.5.25** template<typename \_CharT, typename \_Traits> **basic\_streambuf**< \_CharT, \_Traits > \* **basic\_ios**< \_CharT, \_Traits >::rdbuf ( **basic\_streambuf**< \_CharT, \_Traits > \* \_\_sb )

Changing the underlying buffer.

#### Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.618.5.26** template<typename \_CharT, typename \_Traits> **iosstate** std::basic\_ios< \_CharT, \_Traits >::rdstate ( ) const  
[inline]

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See std::ios\_base::iosstate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.618.5.27** void std::ios\_base::register\_callback ( **event\_callback** \_\_fn, int \_\_index ) [inherited]

Add the callback \_\_fn with parameter \_\_index.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.618.5.28** `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline]`, `[inherited]`

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.618.5.29** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

## Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.618.5.30** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline]`

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file `basic_ios.h`.

Referenced by std::basic\_ostream< char >::M\_write(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.618.5.31** static bool std::ios\_base::sync\_with\_stdio( bool \_\_sync = true ) [static], [inherited]

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.618.5.32** template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie( ) const [inline]

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file `basic_ios.h`.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.618.5.33** template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie( basic\_ostream< \_CharT, \_Traits > \* \_\_tiestr ) [inline]

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic\_ios.h.

**4.618.5.34** void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline],[inherited]

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**4.618.5.35** template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios< \_CharT, \_Traits >::widen ( char \_\_c ) const [inline]

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios< char, \_Traits >::fill(), std::basic\_istream< char >::get(), std::basic\_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

**4.618.5.36** streamsize std::ios\_base::width ( ) const [inline],[inherited]

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::operator>>().

**4.618.5.37 streamsize std::ios\_base::width ( streamsize \_\_wide )** [inline],[inherited]

Changing flags.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

#### Returns

The previous value of width().

Definition at line 653 of file ios\_base.h.

**4.618.5.38 static int std::ios\_base::xalloc ( ) throw ()** [static],[inherited]

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 4.618.6 Member Data Documentation

**4.618.6.1 const fmtflags std::ios\_base::adjustfield** [static],[inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_Outlter >::do\_put(), std::internal(), std::left(), and std::right().

**4.618.6.2 const openmode std::ios\_base::app** [static],[inherited]

Seek to end before each write.

Definition at line 364 of file ios\_base.h.

**4.618.6.3 const openmode std::ios\_base::ate** [static],[inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**4.618.6.4 const iostate std::ios\_base::badbit** [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

**4.618.6.5 const fmtflags std::ios\_base::basefield** [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 313 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.618.6.6 const seekdir std::ios\_base::beg** [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**4.618.6.7 const openmode std::ios\_base::binary** [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**4.618.6.8 const fmtflags std::ios\_base::boolalpha** [static], [inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

**4.618.6.9 const seekdir std::ios\_base::cur** [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**4.618.6.10 const fmtflags std::ios\_base::dec** [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

**4.618.6.11 const seekdir std::ios\_base::end** [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**4.618.6.12 const iostate std::ios\_base::eofbit** [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.618.6.13 const iostate std::ios\_base::failbit** [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.618.6.14 const fmtflags std::ios\_base::fixed** [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

**4.618.6.15 const fmtflags std::ios\_base::floatfield** [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

**4.618.6.16** `const ios_base::goodbit` `[static], [inherited]`

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.618.6.17** `const fmtflags std::ios_base::hex` `[static], [inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.618.6.18** `const openmode std::ios_base::in` `[static], [inherited]`

Open for input. Default for ifstream and fstream.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**4.618.6.19** `const fmtflags std::ios_base::internal` `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

**4.618.6.20** `const fmtflags std::ios_base::left` `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.618.6.21 const fmtflags std::ios\_base::oct** [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.618.6.22 const openmode std::ios\_base::out** [static], [inherited]

Open for output. Default for ofstream and fstream.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**4.618.6.23 const fmtflags std::ios\_base::right** [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios\_base.h.

Referenced by std::right().

**4.618.6.24 const fmtflags std::ios\_base::scientific** [static], [inherited]

Generates floating-point output in scientific notation.

Definition at line 286 of file ios\_base.h.

Referenced by std::scientific().

**4.618.6.25 const fmtflags std::ios\_base::showbase** [static], [inherited]

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

**4.618.6.26 const fmtflags std::ios\_base::showpoint** [static], [inherited]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**4.618.6.27 const fmtflags std::ios\_base::showpos** [static], [inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**4.618.6.28 const fmtflags std::ios\_base::skipws** [static], [inherited]

Skips leading white space before certain input operations.

Definition at line 300 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), and std::skipws().

**4.618.6.29** `const openmode std::ios_base::trunc` [static], [inherited]

Open for input. Default for ofstream.

Definition at line 381 of file ios\_base.h.

**4.618.6.30** `const fmtflags std::ios_base::unitbuf` [static], [inherited]

Flushes output after each output operation.

Definition at line 303 of file ios\_base.h.

Referenced by std::nunitbuf(), std::unitbuf(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

**4.618.6.31** `const fmtflags std::ios_base::uppercase` [static], [inherited]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios\_base.h.

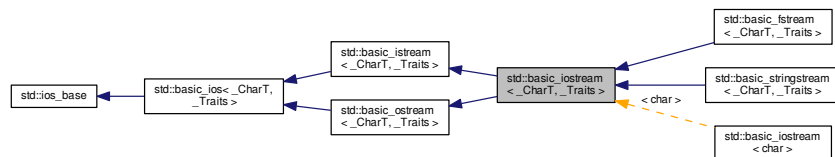
Referenced by std::num\_put< \_CharT, \_Outiter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

## 4.619 std::basic\_istream< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits >:



### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`

- typedef num\_get< \_CharT, istreambuf\_iterator< \_CharT, \_Traits > > \_\_num\_get\_type
- typedef num\_put< \_CharT, ostreambuf\_iterator< \_CharT, \_Traits > > \_\_num\_put\_type
- typedef basic\_ostream< \_CharT, \_Traits > \_\_ostream\_type
- typedef basic\_streambuf< \_CharT, \_Traits > \_\_streambuf\_type
- typedef basic\_streambuf< \_CharT, \_Traits > \_\_streambuf\_type
- typedef \_CharT char\_type
- enum event { erase\_event, imbue\_event, copyfmt\_event }
- typedef void(\* event\_callback)(event \_\_e, ios\_base & \_\_b, int \_\_i)
- typedef \_ios\_Fmtflags fmtflags
- typedef \_Traits::int\_type int\_type
- typedef int io\_state
- typedef \_ios\_istate iostate
- typedef \_Traits::off\_type off\_type
- typedef int open\_mode
- typedef \_ios\_Openmode openmode
- typedef \_Traits::pos\_type pos\_type
- typedef int seek\_dir
- typedef \_ios\_Seekdir seekdir
- typedef std::streamoff streamoff
- typedef std::streampos streampos
- typedef \_Traits traits\_type
  
- typedef num\_put< \_CharT, ostreambuf\_iterator< \_CharT, \_Traits > > \_\_num\_put\_type

#### Public Member Functions

- basic\_istream (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
- virtual ~basic\_istream ()
- const locale & \_M\_getloc () const
- void \_M\_setstate (iostate \_\_state)
- bool bad () const
- void clear (iostate \_\_state=goodbit)
- basic\_ios & copyfmt (const basic\_ios & \_\_rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate \_\_except)
- bool fail () const
- char\_type fill () const
- char\_type fill (char\_type \_\_ch)
- fmtflags flags () const
- fmtflags flags (fmtflags \_\_fmtfl)
- \_\_ostream\_type & flush ()

- [streamsize gcount](#) () const
  - [locale getloc](#) () const
  - [bool good](#) () const
  - [locale imbue](#) (const [locale](#) &\_\_loc)
  - [long & iword](#) (int \_\_ix)
  - [char narrow](#) (char\_type \_\_c, char \_\_default) const
  - [\\_\\_ostream\\_type & operator<<](#) (const void \*\_\_p)
  - [\\_\\_ostream\\_type & operator<<](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [\\_\\_istream\\_type & operator>>](#) (void \*&\_\_p)
  - [\\_\\_istream\\_type & operator>>](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [streamsize precision](#) () const
  - [streamsize precision](#) ([streamsize](#) \_\_prec)
  - [void \\*& pword](#) (int \_\_ix)
  - [basic\\_streambuf< \\_CharT, \\_Traits > \\* rdbuf](#) () const
  - [basic\\_streambuf< \\_CharT, \\_Traits > \\* rdbuf](#) ([basic\\_streambuf< \\_CharT, \\_Traits > \\* \\_\\_sb](#))
  - [iostate rdstate](#) () const
  - [void register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [\\_\\_ostream\\_type & seekp](#) (pos\_type)
  - [\\_\\_ostream\\_type & seekp](#) (off\_type, [ios\\_base::seekdir](#))
  - [fmtflags setf](#) (fmtflags \_\_fmtfl)
  - [fmtflags setf](#) (fmtflags \_\_fmtfl, [fmtflags](#) \_\_mask)
  - [void setstate](#) ([iostate](#) \_\_state)
  - [pos\\_type tellp](#) ()
  - [basic\\_ostream< \\_CharT, \\_Traits > \\* tie](#) () const
  - [basic\\_ostream< \\_CharT, \\_Traits > \\* tie](#) ([basic\\_ostream< \\_CharT, \\_Traits > \\* \\_\\_tiestr](#))
  - [void unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type widen](#) (char \_\_c) const
  - [streamsize width](#) () const
  - [streamsize width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_istream\\_type & operator>>](#) ([\\_\\_istream\\_type](#) &(\_\_pf)(\_\_istream\_type &))
  - [\\_\\_istream\\_type & operator>>](#) ([\\_\\_ios\\_type](#) &(\_\_pf)(\_\_ios\_type &))
  - [\\_\\_istream\\_type & operator>>](#) ([ios\\_base](#) &(\_\_pf)([ios\\_base](#) &))

## Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_istream\\_type & operator>>](#) (bool &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (short &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned short &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (int &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (long long &\_\_n)

- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type` [get](#) ()
- [\\_\\_istream\\_type](#) & [get](#) (`char_type` &\_\_c)
- [\\_\\_istream\\_type](#) & [get](#) (`char_type` \*\_\_s, [streamsize](#) \_\_n, `char_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) (`char_type` \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, `char_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
- [\\_\\_istream\\_type](#) & [getline](#) (`char_type` \*\_\_s, [streamsize](#) \_\_n, `char_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [getline](#) (`char_type` \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, `int_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ()
- `int_type` [peek](#) ()
- [\\_\\_istream\\_type](#) & [read](#) (`char_type` \*\_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [readsome](#) (`char_type` \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [putback](#) (`char_type` \_\_c)
- [\\_\\_istream\\_type](#) & [unget](#) ()
- `int` [sync](#) ()
- `pos_type` [tellg](#) ()
- [\\_\\_istream\\_type](#) & [seekg](#) (`pos_type`)
- [\\_\\_istream\\_type](#) & [seekg](#) (`off_type`, `ios_base::seekdir`)
- [operator void \\*](#) () const
- `bool` [operator!](#) () const
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) (`ios_base` &(\*\_\_pf)(`ios_base` &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)

- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`

- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

#### Protected Types

- enum { **\_S\_local\_word\_size** }

#### Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)

#### Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.619.1 Detailed Description

```
template<typename _CharT, typename _Traits>class std::basic_iostream< _CharT, _Traits >
```

Template class basic\_iostream.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 795 of file istream.

## 4.619.2 Member Typedef Documentation

```
4.619.2.1 template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 88 of file basic\_ios.h.

```
4.619.2.2 typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the ios_base object.                      |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 436 of file ios\_base.h.

```
4.619.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`

- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.619.2.4 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 330 of file `ios_base.h`.

#### 4.619.2.5 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 361 of file `ios_base.h`.

#### 4.619.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK\_CUR in the C standard library.
- end, equivalent to SEEK\_END in the C standard library.

Definition at line 393 of file ios\_base.h.

### 4.619.3 Member Enumeration Documentation

#### 4.619.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

Definition at line 419 of file ios\_base.h.

### 4.619.4 Constructor & Destructor Documentation

#### 4.619.4.1 template<typename \_CharT, typename \_Traits> std::basic\_iostream< \_CharT, \_Traits >::basic\_iostream ( basic\_streambuf< \_CharT, \_Traits > \* \_\_sb ) [inline],[explicit]

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 820 of file istream.

#### 4.619.4.2 template<typename \_CharT, typename \_Traits> virtual std::basic\_iostream< \_CharT, \_Traits >::~~basic\_iostream ( ) [inline],[virtual]

Destructor does nothing.

Definition at line 827 of file istream.

### 4.619.5 Member Function Documentation

#### 4.619.5.1 const locale& std::ios\_base::M.getloc ( ) const [inline],[inherited]

Locale access.

#### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::time\_put< \_CharT, \_Outlter >::put().

**4.619.5.2** `template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::M_write ( const char_type * __s, streamsize __n ) [inline], [inherited]`

Core write functionality, without sentry.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 311 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

**4.619.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

**4.619.5.4** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.619.5.5** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of \_\_rhs into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

## Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, `std::tie()`, and `std::ios_base::width()`.

**4.619.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const` `[inline]`, `[inherited]`

Fast error checking.

## Returns

True if the eofbit is set.

Note that other `iosstate` flags may also be set.

Definition at line 184 of file `basic_ios.h`.

**4.619.5.7** `template<typename _CharT, typename _Traits> iosstate std::basic_ios< _CharT, _Traits >::exceptions ( ) const` `[inline]`, `[inherited]`

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iosstate)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**4.619.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iosstate __except )` `[inline]`, `[inherited]`

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

**4.619.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const` `[inline]`, `[inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**4.619.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const` `[inline]`, `[inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 364 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::basic\_ios< char, \_Traits >::fill().

**4.619.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )` `[inline]`, `[inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file basic\_ios.h.

**4.619.5.12** `fmtflags std::ios_base::flags ( ) const` `[inline],[inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.619.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline],[inherited]`

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file ios\_base.h.

**4.619.5.14** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::flush ( )` `[inherited]`

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( )->pubsync ( )`, and if that returns -1, sets `badbit`.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

4.619.5.15 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( )  
const [inline],[inherited]`

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.619.5.16 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream<  
_CharT, _Traits >::get ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.619.5.17 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT,  
_Traits >::get ( char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

#### Returns

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.619.5.18 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT,  
_Traits >::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.619.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline], [inherited]`

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

## Returns

\*this

Returns `get(__s, __n, widen("\n"))`.

Definition at line 354 of file istream.

**4.619.5.20** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

## Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

## Returns

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::putc()`.

**4.619.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline], [inherited]`

Extraction into another streambuf.

## Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

## Returns

\*this

Returns `get(__sb, widen("\n"))`.

Definition at line 387 of file istream.

**4.619.5.22** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::sngetc()`.

**4.619.5.23** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline], [inherited]`

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s, __n, widen("\n"))`.

Definition at line 427 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**4.619.5.24** `locale std::ios_base::getloc ( ) const [inline], [inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, `std::operator>>()`, and `std::ws()`.

4.619.5.25 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const`  
`[inline], [inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.619.5.26 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim )` `[inherited]`

Discarding characters.

#### Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

#### Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snexctc()`.

4.619.5.27 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n )` `[inherited]`

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.619.5.28** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::ignore( void )` [inherited]

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.619.5.29** `template<typename _CharT, typename _Traits> locale basic_ios< _CharT, _Traits>::imbue( const locale & __loc )` [inherited]

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file basic\_ios.tcc.

References std::ios\_base::imbue().

Referenced by std::operator<<().

**4.619.5.30** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits>::init( basic_streambuf< _CharT, _Traits> * __sb )` [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_ios< char, \_Traits >::basic\_ios(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >::basic\_ostreamstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

**4.619.5.31** long& std::ios\_base::iword ( int \_\_ix ) [inline],[inherited]

Access to integer array.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

#### Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios\_base.h.

**4.619.5.32** template<typename \_CharT, typename \_Traits> char std::basic\_ios< \_CharT, \_Traits >::narrow ( char\_type \_\_c, char \_\_dfault ) const [inline],[inherited]

Squeezes characters.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file basic\_ios.h.

**4.619.5.33** template<typename \_CharT, typename \_Traits> std::basic\_ios< \_CharT, \_Traits >::operator void \* ( ) const [inline],[inherited]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic\_ios.h.

4.619.5.34 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 119 of file `basic_ios.h`.

4.619.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ostream_type &(*)(__ostream_type &)__pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

4.619.5.36 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ios_type &(*)(__ios_type &)__pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

4.619.5.37 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( ios_base &(*)(ios_base &)__pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

4.619.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

4.619.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned long __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

**4.619.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( bool __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

**4.619.5.41** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<<( short __n ) [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**4.619.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( unsigned short __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 181 of file ostream.

**4.619.5.43** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( int __n ) [inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios\_base::basefield, std::ios\_base::flags(), std::ios\_base::hex, and std::ios\_base::oct.

**4.619.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( unsigned int __n ) [inline],[inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 192 of file ostream.

**4.619.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( long long __n ) [inline],[inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 201 of file ostream.

```
4.619.5.46 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits
>::operator<<(unsigned long long __n) [inline],[inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

```
4.619.5.47 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits
>::operator<<(double __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

```
4.619.5.48 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits
>::operator<<(float __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

```
4.619.5.49 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits
>::operator<<(long double __f) [inline],[inherited]
```

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**4.619.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( const void * __p ) [inline], [inherited]`

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

**4.619.5.51** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.619.5.52 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file `istream`.

4.619.5.53 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 124 of file `istream`.

4.619.5.54 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

4.619.5.55 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( bool & __n ) [inline], [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

4.619.5.56 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( short & __n ) [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.619.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned short & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.619.5.58** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.619.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned int & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 182 of file istream.

**4.619.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 186 of file istream.

**4.619.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 190 of file istream.

**4.619.5.62** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 195 of file istream.

4.619.5.63 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long long & __n ) [inline],[inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

4.619.5.64 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( float & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

4.619.5.65 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( double & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

4.619.5.66 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long double & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**4.619.5.67** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( void *& __p ) [inline], [inherited]`

Basic arithmetic extractors.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.619.5.68** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, `failbit` is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.619.5.69 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::int_type basic_istream< _CharT, _Traits>::peek( void ) [inherited]`

Looking ahead in the stream.

#### Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.619.5.70 `streamsize std::ios_base::precision( ) const [inline],[inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

4.619.5.71 `streamsize std::ios_base::precision( streamsize __prec ) [inline],[inherited]`

Changing flags.

#### Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

#### Returns

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

4.619.5.72 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::put( char_type __c ) [inherited]`

Simple insertion.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_istream< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**4.619.5.73** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::putback ( char_type __c ) [inherited]`

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**4.619.5.74** `void*& std::ios_base::pword ( int __ix ) [inline], [inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

4.619.5.75 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf( ) const` [inline], [inherited]

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

4.619.5.76 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* basic_ios<_CharT, _Traits>::rdbuf( basic_streambuf<_CharT, _Traits>* __sb )` [inherited]

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

4.619.5.77 `template<typename _CharT, typename _Traits> istate std::basic_ios< _CharT, _Traits >::rdstate ( ) const`  
`[inline], [inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::istate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

4.619.5.78 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::read ( char_type * __s, streamsize __n )` `[inherited]`

Extraction without delimiters.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

#### Returns

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.619.5.79 `template<typename _CharT, typename _Traits> streamsize basic_istream< _CharT, _Traits >::readsome ( char_type * __s, streamsize __n )` `[inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.619.5.80** void `std::ios_base::register_callback ( event_callback __fn, int __index )` [inherited]

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.619.5.81** template<typename `_CharT`, typename `_Traits`> `basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::seekg ( pos_type __pos )` [inherited]

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos (__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.619.5.82 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.619.5.83 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::seekp ( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.619.5.84 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.619.5.85** `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline]`, `[inherited]`

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.619.5.86** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

**Parameters**

|                      |                                           |
|----------------------|-------------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                  |
| <code>__mask</code>  | The flags mask for <code>__fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `__fmtfl & mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.619.5.87** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline]`, `[inherited]`

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.619.5.88** template<typename \_CharT, typename \_Traits> int basic\_istream< \_CharT, \_Traits >::sync ( void )  
[inherited]

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_streambuf< \_CharT, \_Traits >::pubsync(), std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.619.5.89** static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true ) [static], [inherited]

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.619.5.90** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg( void ) [inherited]`

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**4.619.5.91** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp( ) [inherited]`

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**4.619.5.92** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits >*& std::basic_ios< _CharT, _Traits >::tie( ) const [inline], [inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.619.5.93 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]`

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 301 of file `basic_ios.h`.

4.619.5.94 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::unget ( void ) [inherited]`

Unextracting the previous character.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

4.619.5.95 `void std::ios_base::unsetf ( fmtflags __mask ) [inline], [inherited]`

Clearing format flags.

#### Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

4.619.5.96 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen ( char __c )`  
`const [inline],[inherited]`

Widens characters.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios< char, _Traits >::fill()`, `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::getline()`, `std::tr2::operator>>()`, and `std::operator>>()`.

4.619.5.97 `streamsize std::ios_base::width ( ) const [inline],[inherited]`

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outlter >::do_put()`, and `std::operator>>()`.

4.619.5.98 `streamsize std::ios_base::width ( streamsize __wide ) [inline],[inherited]`

Changing flags.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of width().

Definition at line 653 of file ios\_base.h.

**4.619.5.99** template<typename \_CharT, typename \_Traits > **basic\_ostream**< \_CharT, \_Traits > & **basic\_ostream**< \_CharT, \_Traits >::write ( const char\_type \* \_\_s, streamsize \_\_n ) [inherited]

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**4.619.5.100** static int **std::ios\_base::xalloc** ( ) throw () [static], [inherited]

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.619.6 Member Data Documentation**

**4.619.6.1** template<typename \_CharT, typename \_Traits> streamsize **std::basic\_istream**< \_CharT, \_Traits >::\_M\_gcount [protected], [inherited]

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file istream.

Referenced by std::basic\_istream< char >::gcount(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::basic\_istream< char >::~~basic\_istream().

#### 4.619.6.2 const fmtflags std::ios\_base::adjustfield [static],[inherited]

A mask of left|right|internal. Useful for the 2-arg form of setf.

Definition at line 310 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::internal(), std::left(), and std::right().

#### 4.619.6.3 const openmode std::ios\_base::app [static],[inherited]

Seek to end before each write.

Definition at line 364 of file ios\_base.h.

#### 4.619.6.4 const openmode std::ios\_base::ate [static],[inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

#### 4.619.6.5 const iostate std::ios\_base::badbit [static],[inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::M\_write(), std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~~sentry().

#### 4.619.6.6 const fmtflags std::ios\_base::basefield [static],[inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 313 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 4.619.6.7 const seekdir std::ios\_base::beg [static],[inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

#### 4.619.6.8 const openmode std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 4.619.6.9 const fmtflags std::ios\_base::boolalpha [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

#### 4.619.6.10 const seekdir std::ios\_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 4.619.6.11 const fmtflags std::ios\_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

#### 4.619.6.12 const seekdir std::ios\_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.619.6.13 const iostate std::ios\_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and

std::ws().

#### 4.619.6.14 const iostate std::ios\_base::failbit [static],[inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.619.6.15 const fmtflags std::ios\_base::fixed [static],[inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.619.6.16 const fmtflags std::ios\_base::floatfield [static],[inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.619.6.17 const iostate std::ios\_base::goodbit [static],[inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 4.619.6.18 const fmtflags std::ios\_base::hex [static],[inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.619.6.19 const openmode std::ios\_base::in** [static],[inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**4.619.6.20 const fmtflags std::ios\_base::internal** [static],[inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

**4.619.6.21 const fmtflags std::ios\_base::left** [static],[inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, and `std::left()`.

**4.619.6.22 const fmtflags std::ios\_base::oct** [static],[inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**4.619.6.23 const openmode std::ios\_base::out** [static],[inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**4.619.6.24 const fmtflags std::ios\_base::right** [static],[inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.619.6.25 const fmtflags std::ios\_base::scientific** [static],[inherited]

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

#### 4.619.6.26 `const fmtflags std::ios_base::showbase` `[static]`, `[inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

#### 4.619.6.27 `const fmtflags std::ios_base::showpoint` `[static]`, `[inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

#### 4.619.6.28 `const fmtflags std::ios_base::showpos` `[static]`, `[inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

#### 4.619.6.29 `const fmtflags std::ios_base::skipws` `[static]`, `[inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

#### 4.619.6.30 `const openmode std::ios_base::trunc` `[static]`, `[inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

#### 4.619.6.31 `const fmtflags std::ios_base::unitbuf` `[static]`, `[inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

#### 4.619.6.32 `const fmtflags std::ios_base::uppercase` `[static]`, `[inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

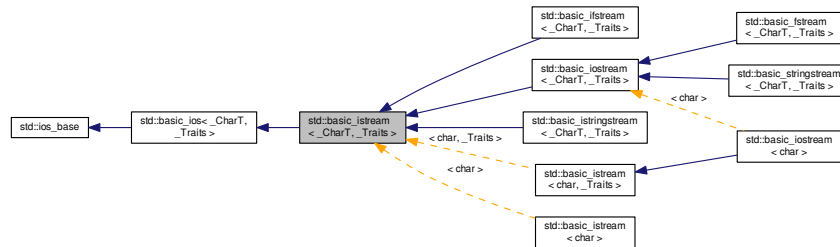
Definition at line 307 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [istream](#)

Inheritance diagram for `std::basic_istream<_CharT, _Traits>`:



- class `sentry`  
*Performs setup work for input streams.*

- typedef **ctype**< \_CharT > **\_\_ctype\_type**
- typedef **basic\_ios**< \_CharT,  
\_Traits > **\_\_ios\_type**
- typedef **basic\_istream**< \_CharT,  
\_Traits > **\_\_istream\_type**
- typedef **num\_get**< \_CharT,  
**istreambuf\_iterator**< \_CharT,  
\_Traits > > **\_\_num\_get\_type**
- typedef **basic\_streambuf**  
< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum **event** { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
- typedef void(\* **event\_callback**)(**event** \_\_e, **ios\_base** &\_\_b, int \_\_i)
- typedef \_Ios\_Fmtflags **fmtflags**
- typedef \_Traits::int\_type **int\_type**
- typedef int **io\_state**
- typedef \_Ios\_ostate **iostate**
- typedef \_Traits::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_Ios\_Openmode **openmode**
- typedef \_Traits::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_Ios\_Seekdir **seekdir**
- typedef **std::streamoff** **streamoff**
- typedef **std::streampos** **streampos**
- typedef \_Traits **traits\_type**

- typedef num\_put< \_CharT, ostreambuf\_iterator< \_CharT, \_Traits > > \_\_num\_put\_type

#### Public Member Functions

- basic\_istream ( \_\_streambuf\_type \* \_\_sb)
  - virtual ~basic\_istream ()
  - const locale & \_M\_getloc () const
  - void \_M\_setstate (iostate \_\_state)
  - bool bad () const
  - void clear (iostate \_\_state=goodbit)
  - basic\_ios & copyfmt (const basic\_ios & \_\_rhs)
  - bool eof () const
  - iostate exceptions () const
  - void exceptions (iostate \_\_except)
  - bool fail () const
  - char\_type fill () const
  - char\_type fill (char\_type \_\_ch)
  - fmtflags flags () const
  - fmtflags flags (fmtflags \_\_fmtfl)
  - streamsize gcount () const
  - locale getloc () const
  - bool good () const
  - locale imbue (const locale & \_\_loc)
  - long & iword (int \_\_ix)
  - char narrow (char\_type \_\_c, char \_\_dfault) const
  - \_\_istream\_type & operator>> (void \*& \_\_p)
  - \_\_istream\_type & operator>> ( \_\_streambuf\_type \* \_\_sb)
  - streamsize precision () const
  - streamsize precision (streamsize \_\_prec)
  - void \*& pword (int \_\_ix)
  - basic\_streambuf< \_CharT, \_Traits > \* rdbuf () const
  - basic\_streambuf< \_CharT, \_Traits > \* rdbuf (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
  - iostate rdstate () const
  - void register\_callback (event\_callback \_\_fn, int \_\_index)
  - fmtflags setf (fmtflags \_\_fmtfl)
  - fmtflags setf (fmtflags \_\_fmtfl, fmtflags \_\_mask)
  - void setstate (iostate \_\_state)
  - basic\_ostream< \_CharT, \_Traits > \* tie () const
  - basic\_ostream< \_CharT, \_Traits > \* tie (basic\_ostream< \_CharT, \_Traits > \* \_\_tiestr)
  - void unsetf (fmtflags \_\_mask)
  - char\_type widen (char \_\_c) const
  - streamsize width () const
  - streamsize width (streamsize \_\_wide)
- 
- \_\_istream\_type & operator>> ( \_\_istream\_type & (\*\_\_pf)(\_\_istream\_type &))
  - \_\_istream\_type & operator>> ( \_\_ios\_type & (\*\_\_pf)(\_\_ios\_type &))

- [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the *operator>>* functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_istream\\_type](#) & [operator>>](#) ([bool](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([short](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned short](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([int](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned int](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([long](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned long](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([long long](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([unsigned long long](#) &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([float](#) &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([double](#) &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([long double](#) &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [int\\_type](#) [get](#) ()
- [\\_\\_istream\\_type](#) & [get](#) ([char\\_type](#) &\_\_c)
- [\\_\\_istream\\_type](#) & [get](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
- [\\_\\_istream\\_type](#) & [getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ()
- [int\\_type](#) [peek](#) ()
- [\\_\\_istream\\_type](#) & [read](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [readsome](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [putback](#) ([char\\_type](#) \_\_c)
- [\\_\\_istream\\_type](#) & [unget](#) ()
- [int](#) [sync](#) ()
- [pos\\_type](#) [tellg](#) ()
- [\\_\\_istream\\_type](#) & [seekg](#) ([pos\\_type](#))
- [\\_\\_istream\\_type](#) & [seekg](#) ([off\\_type](#), [ios\\_base::seekdir](#))
- [operator void \\*](#) () const
- [bool](#) [operator!](#) () const

## Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

## Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

## Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

## Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT >  
[\\_istream\\_type](#) & [\\_M\\_extract](#) (\_ValueT & \_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [init](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

## 4.620.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream<_CharT, _Traits>
```

Template class `basic_istream`.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.

Definition at line 58 of file `istream`.

## 4.620.2 Member Typedef Documentation

```
4.620.2.1 template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.620.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i)` [inherited]

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.620.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.620.2.4 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 330 of file `ios_base.h`.

#### 4.620.2.5 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 361 of file `ios_base.h`.

#### 4.620.2.6 `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

### 4.620.3 Member Enumeration Documentation

#### 4.620.3.1 `enum std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

## 4.620.4 Constructor &amp; Destructor Documentation

4.620.4.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::basic_istream ( __streambuf_type* __sb ) [inline],[explicit]`

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 93 of file istream.

4.620.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_istream< _CharT, _Traits >::~basic_istream ( ) [inline],[virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 103 of file istream.

## 4.620.5 Member Function Documentation

4.620.5.1 `const locale& std::ios_base::M_getloc ( ) const [inline],[inherited]`

Locale access.

## Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, and `std::time_put< _CharT, _Outlter >::put()`.

4.620.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline],[inherited]`

Fast error checking.

## Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

4.620.5.3 `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

## Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.620.5.4** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits> & basic_ios< _CharT, _Traits>::copyfmt ( const basic_ios< _CharT, _Traits> & __rhs ) [inherited]`

Copies fields of \_\_rhs into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

## Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

References std::basic\_ios< \_CharT, \_Traits >::exceptions(), std::basic\_ios< \_CharT, \_Traits >::fill(), std::ios\_base::flags(), std::ios\_base::getloc(), std::ios\_base::precision(), std::basic\_ios< \_CharT, \_Traits >::tie(), std::tie(), and std::ios\_base::width().

**4.620.5.5** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits>::eof ( ) const [inline], [inherited]`

Fast error checking.

## Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic\_ios.h.

**4.620.5.6** `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits>::exceptions ( ) const [inline], [inherited]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt().

**4.620.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except ) [inline],[inherited]`

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

**4.620.5.8** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline],[inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**4.620.5.9** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const`  
`[inline], [inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 364 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

**4.620.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )`  
`[inline], [inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file `basic_ios.h`.

**4.620.5.11** `fmtflags std::ios_base::flags ( ) const` `[inline], [inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.620.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline], [inherited]`

Setting new format flags all at once.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file ios\_base.h.

**4.620.5.13** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( )`  
`const [inline]`

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

**4.620.5.14** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get ( void )`

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.620.5.15** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( char_type & __c )`

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.620.5.16 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim )`

Simple multiple-character extraction.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

#### Returns

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

4.620.5.17 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline]`

Simple multiple-character extraction.

#### Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

4.620.5.18 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim )`

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::setstate()`, `std::basic_streambuf< _CharT, _Traits>::sgetc()`, `std::basic_streambuf< _CharT, _Traits>::snextc()`, and `std::basic_streambuf< _CharT, _Traits>::sputc()`.

4.620.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::get ( __streambuf_type & __sb ) [inline]`

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

4.620.5.20 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::getline( char_type * __s, streamsize __n, char_type __delim )`

String extraction.

#### Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

#### Returns

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_streambuf< _CharT, _Traits>::sbumpc()`, `std::basic_ios< _CharT, _Traits>::setstate()`, `std::basic_streambuf< _CharT, _Traits>::sgetc()`, and `std::basic_streambuf< _CharT, _Traits>::snextc()`.

4.620.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::getline( char_type * __s, streamsize __n ) [inline]`

String extraction.

#### Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

#### Returns

`*this`

Returns `getline(__s, __n, widen('\n'))`.

Definition at line 427 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**4.620.5.22** locale std::ios\_base::getloc ( ) const [inline],[inherited]

Locale access.

**Returns**

A copy of the current locale.

If imbue(loc) has previously been called, then this function returns loc. Otherwise, it returns a copy of std::locale(), the global C++ locale.

Definition at line 695 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::money\_put< \_CharT, \_Outiter >::do\_put(), std::operator>>(), and std::ws().

**4.620.5.23** template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::good ( ) const [inline],[inherited]

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic\_ios.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.620.5.24** template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::ignore ( streamsize \_\_n, int\_type \_\_delim )

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.620.5.25** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::ignore ( streamsize __n )`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.620.5.26** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::ignore ( void )`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.620.5.27** `template<typename _CharT, typename _Traits> locale basic_ios< _CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file basic\_ios.tcc.

References std::ios\_base::imbue().

Referenced by std::operator<<().

**4.620.5.28** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init ( basic_streambuf< _CharT, _Traits > * __sb )` `[protected]`, `[inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_ios< char, \_Traits >::basic\_ios(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >::basic\_ostreamstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

**4.620.5.29** `long& std::ios_base::iword ( int __ix )` `[inline]`, `[inherited]`

Access to integer array.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

#### Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios\_base.h.

**4.620.5.30** `template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow ( char_type __c, char __dfault ) const` `[inline]`, `[inherited]`

Squeezes characters.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file basic\_ios.h.

**4.620.5.31** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const`  
`[inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic\_ios.h.

**4.620.5.32** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const`  
`[inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic\_ios.h.

**4.620.5.33** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __istream_type &(*)(__istream_type &) __pf )` `[inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file istream.

**4.620.5.34** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __ios_type &(*)(__ios_type &) __pf )` `[inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 124 of file istream.

**4.620.5.35** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*)(ios_base &) __pf )` `[inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file istream.

**4.620.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( bool &__n )` `[inline]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

**4.620.5.37** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::operator>> ( short & __n )`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

**4.620.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( unsigned short & __n ) [inline]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.620.5.39** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::operator>> ( int & __n )`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.620.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned int & __n ) [inline]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.620.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long & __n ) [inline]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**4.620.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long & __n ) [inline]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**4.620.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long long & __n ) [inline]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**4.620.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long long & __n ) [inline]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**4.620.5.45** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( float & __f ) [inline]`

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

4.620.5.46 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( double & __f ) [inline]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

4.620.5.47 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( long double & __f ) [inline]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

4.620.5.48 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( void *& __p ) [inline]`

Basic arithmetic extractors.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

4.620.5.49 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::operator>> ( __streambuf_type * __sb )`

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.620.5.50** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::peek ( void )`

Looking ahead in the stream.

## Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.620.5.51** `streamsize std::ios_base::precision ( ) const` `[inline]`, `[inherited]`

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**4.620.5.52** `streamsize std::ios_base::precision ( streamsize __prec )` `[inline]`, `[inherited]`

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

**4.620.5.53** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::putback ( char_type __c )`

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::rdstate()`, `std::basic_ios< _CharT, _Traits>::setstate()`, and `std::basic_streambuf< _CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

**4.620.5.54** `void*& std::ios_base::pword ( int __ix )` `[inline]`, `[inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

**4.620.5.55** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::rdbuf ( ) const` `[inline], [inherited]`

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.620.5.56** `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf ( basic_streambuf< _CharT, _Traits > * __sb )` `[inherited]`

Changing the underlying buffer.

#### Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.620.5.57** `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const` `[inline], [inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See std::ios\_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.620.5.58** template<typename \_CharT, typename \_Traits > **basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::read ( char\_type \* \_\_s, streamsize \_\_n )**

Extraction without delimiters.

**Parameters**

|            |                                        |
|------------|----------------------------------------|
| <b>__s</b> | A character array.                     |
| <b>__n</b> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.620.5.59** template<typename \_CharT, typename \_Traits > **streamsize basic\_istream< \_CharT, \_Traits >::readsome ( char\_type \* \_\_s, streamsize \_\_n )**

Extraction until the buffer is exhausted, but no more.

**Parameters**

|            |                                        |
|------------|----------------------------------------|
| <b>__s</b> | A character array.                     |
| <b>__n</b> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.620.5.60** void `std::ios_base::register_callback ( event_callback __fn, int __index )` [inherited]

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.620.5.61** template<typename `_CharT`, typename `_Traits`> `basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::seekg ( pos_type __pos )`

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos (__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.620.5.62 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::seekg ( off_type __off, ios_base::seekdir __dir )`

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::rdstate()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.620.5.63 `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline]`, `[inherited]`

Setting new format flags.

#### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

4.620.5.64 `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

#### Parameters

|                      |                                           |
|----------------------|-------------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                  |
| <code>__mask</code>  | The flags mask for <code>__fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.620.5.65** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )`  
`[inline], [inherited]`

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file `basic_ios.h`.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::tr2::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**4.620.5.66** `template<typename _CharT, typename _Traits> int basic_istream< _CharT, _Traits >::sync ( void )`

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.620.5.67 `static bool std::ios_base::sync_with_stdio ( bool __sync = true ) [static],[inherited]`

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

4.620.5.68 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg ( void )`

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 817 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

4.620.5.69 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline],[inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.620.5.70 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( basic_ostream< _CharT, _Traits >* __tiestr ) [inline],[inherited]`

Ties this stream to an output stream.

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 301 of file `basic_ios.h`.

**4.620.5.71** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::unget ( void )`

Unextracting the previous character.

## Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

## Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::rdstate()`, `std::basic_ios< _CharT, _Traits>::setstate()`, and `std::basic_streambuf< _CharT, _Traits>::sungetc()`.

**4.620.5.72** `void std::ios_base::unsetf ( fmtflags __mask ) [inline], [inherited]`

Clearing format flags.

## Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**4.620.5.73** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits>::widen ( char __c ) const [inline], [inherited]`

Widens characters.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios< char, _Traits >::fill()`, `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::getline()`, `std::tr2::operator>>()`, and `std::operator>>()`.

**4.620.5.74** `streamsize std::ios_base::width ( ) const` `[inline]`, `[inherited]`

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**4.620.5.75** `streamsize std::ios_base::width ( streamsize __wide )` `[inline]`, `[inherited]`

Changing flags.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

## Returns

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

**4.620.5.76** `static int std::ios_base::xalloc ( ) throw ()` `[static]`, `[inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.620.6 Member Data Documentation****4.620.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::_M_gcount` `[protected]`**

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream<char>::gcount()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::basic_istream<char>::~~basic_istream()`.

**4.620.6.2 `const fmtflags std::ios_base::adjustfield` `[static]`, `[inherited]`**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outlter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.620.6.3 `const openmode std::ios_base::app` `[static]`, `[inherited]`**

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.620.6.4 `const openmode std::ios_base::ate` `[static]`, `[inherited]`**

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**4.620.6.5 `const iostate std::ios_base::badbit` `[static]`, `[inherited]`**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std-`

std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

#### 4.620.6.6 const fmtflags std::ios\_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 313 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 4.620.6.7 const seekdir std::ios\_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

#### 4.620.6.8 const openmode std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 4.620.6.9 const fmtflags std::ios\_base::boolalpha [static], [inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

#### 4.620.6.10 const seekdir std::ios\_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 4.620.6.11 const fmtflags std::ios\_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

#### 4.620.6.12 const seekdir std::ios\_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.620.6.13 const iostate std::ios\_base::eofbit [static],[inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 4.620.6.14 const iostate std::ios\_base::failbit [static],[inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.620.6.15 const fmtflags std::ios\_base::fixed [static],[inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.620.6.16 const fmtflags std::ios\_base::floatfield [static],[inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.620.6.17 const iostate std::ios\_base::goodbit [static],[inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits

>::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 4.620.6.18 const fmtflags std::ios\_base::hex [static],[inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 4.620.6.19 const openmode std::ios\_base::in [static],[inherited]

Open for input. Default for ifstream and fstream.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

#### 4.620.6.20 const fmtflags std::ios\_base::internal [static],[inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

#### 4.620.6.21 const fmtflags std::ios\_base::left [static],[inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

#### 4.620.6.22 const fmtflags std::ios\_base::oct [static],[inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 4.620.6.23 const openmode std::ios\_base::out [static],[inherited]

Open for output. Default for ofstream and fstream.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(),

`std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

#### 4.620.6.24 `const fmtflags std::ios_base::right` `[static]`, `[inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

#### 4.620.6.25 `const fmtflags std::ios_base::scientific` `[static]`, `[inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

#### 4.620.6.26 `const fmtflags std::ios_base::showbase` `[static]`, `[inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

#### 4.620.6.27 `const fmtflags std::ios_base::showpoint` `[static]`, `[inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

#### 4.620.6.28 `const fmtflags std::ios_base::showpos` `[static]`, `[inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

#### 4.620.6.29 `const fmtflags std::ios_base::skipws` `[static]`, `[inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

#### 4.620.6.30 `const openmode std::ios_base::trunc` `[static]`, `[inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

#### 4.620.6.31 `const fmtflags std::ios_base::unitbuf` `[static]`, `[inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~~sentry()`.

#### 4.620.6.32 `const fmtflags std::ios_base::uppercase` `[static]`, `[inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

### 4.621 `std::basic_istream<_CharT, _Traits>::sentry` Class Reference

#### Public Types

- typedef `__istream_type::__ctype_type` `__ctype_type`
- typedef `_Traits::int_type` `__int_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_Traits` `traits_type`

#### Public Member Functions

- [sentry](#) (`basic_istream<_CharT, _Traits> &__is`, `bool __noskipws=false`)
- [operator bool](#) () const

#### 4.621.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_istream<_CharT, _Traits>::sentry`

Performs setup work for input streams.

Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 657 of file `istream`.

#### 4.621.2 Member Typedef Documentation

4.621.2.1 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream<_CharT, _Traits>::sentry::traits_type`

Easy access to dependent types.

Definition at line 664 of file `istream`.

## 4.621.3 Constructor &amp; Destructor Documentation

4.621.3.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::sentry ( basic_istream< _CharT, _Traits > & __is, bool __noskipws = false ) [explicit]`

The constructor performs all the work.

## Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>__is</code>       | The input stream to guard.            |
| <code>__noskipws</code> | Whether to consume whitespace or not. |

If the stream state is good (`__is.good()` is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if `__noskipws` is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, `std::basic_ios< _CharT, _Traits >::good()`, `std::ios_base::goodbit`, `std::__ctype_abstract_base< _CharT >::is()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::ios_base::skipws`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_ios< _CharT, _Traits >::tie()`.

## 4.621.4 Member Function Documentation

4.621.4.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::operator bool ( ) const [inline], [explicit]`

Quick status checking.

## Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

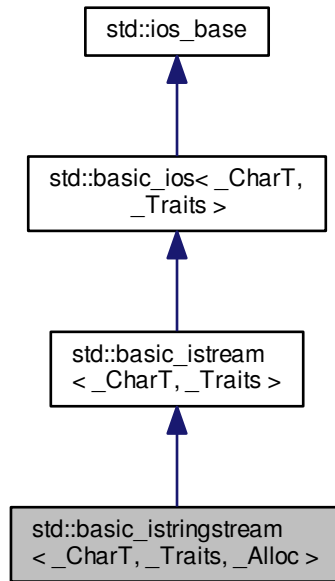
Definition at line 705 of file `istream`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 4.622 std::basic\_istream&lt; \_CharT, \_Traits, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event __e, ios_base & __b, int __i)`
- typedef `_ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`

- typedef int **io\_state**
- typedef \_ios\_istate **iostate**
- typedef traits\_type::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_ios\_Openmode **openmode**
- typedef traits\_type::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_ios\_Seekdir **seekdir**
- typedef **std::streamoff** **streamoff**
- typedef **std::streampos** **streampos**
- typedef \_Traits **traits\_type**
- typedef **num\_put**< \_CharT,  
    **ostreambuf\_iterator**< \_CharT,  
    \_Traits > > **\_\_num\_put\_type**

#### Public Member Functions

- **basic\_istream** (**ios\_base::openmode** \_\_mode=**ios\_base::in**)
- **basic\_istream** (const **\_\_string\_type** &\_\_str, **ios\_base::openmode** \_\_mode=**ios\_base::in**)
- **~basic\_istream** ()
- const **locale** & **\_M\_getloc** () const
- void **\_M\_setstate** (**iostate** \_\_state)
- bool **bad** () const
- void **clear** (**iostate** \_\_state=**goodbit**)
- **basic\_ios** & **copyfmt** (const **basic\_ios** &\_\_rhs)
- bool **eof** () const
- **iostate** **exceptions** () const
- void **exceptions** (**iostate** \_\_except)
- bool **fail** () const
- char\_type **fill** () const
- char\_type **fill** (char\_type \_\_ch)
- **fmtflags** **flags** () const
- **fmtflags** **flags** (**fmtflags** \_\_fmtfl)
- **streamsize** **gcount** () const
- **locale** **getloc** () const
- bool **good** () const
- **locale** **imbue** (const **locale** &\_\_loc)
- long & **word** (int \_\_ix)
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- **\_\_istream\_type** & **operator>>** (void \*&\_\_p)
- **\_\_istream\_type** & **operator>>** (**\_\_streambuf\_type** \*\_\_sb)
- **streamsize** **precision** () const
- **streamsize** **precision** (**streamsize** \_\_prec)
- void \*& **pword** (int \_\_ix)
- **basic\_streambuf**< \_CharT,  
    \_Traits > \* **rdbuf** (**basic\_streambuf**< \_CharT, \_Traits > \*\_\_sb)
- **\_\_stringbuf\_type** \* **rdbuf** () const
- **iostate** **rdstate** () const
- void **register\_callback** (**event\_callback** \_\_fn, int \_\_index)

- `fmtflags` `setf` (`fmtflags` \_\_fmtfl)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
- `void` `setstate` (`iosstate` \_\_state)
- `__string_type` `str` () const
- `void` `str` (const `__string_type` &\_\_s)
- `basic_ostream`< `_CharT`, `_Traits` > \* `tie` () const
- `basic_ostream`< `_CharT`, `_Traits` > \* `tie` (`basic_ostream`< `_CharT`, `_Traits` > \*\_\_tiestr)
- `void` `unsetf` (`fmtflags` \_\_mask)
- `char_type` `widen` (`char` \_\_c) const
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` \_\_wide)
- `__istream_type` & `operator>>` (`__istream_type` &(\_\_pf)(\_\_istream\_type &))
- `__istream_type` & `operator>>` (`__ios_type` &(\_\_pf)(\_\_ios\_type &))
- `__istream_type` & `operator>>` (`ios_base` &(\_\_pf)(`ios_base` &))

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type` & `operator>>` (`bool` &\_\_n)
- `__istream_type` & `operator>>` (`short` &\_\_n)
- `__istream_type` & `operator>>` (`unsigned short` &\_\_n)
- `__istream_type` & `operator>>` (`int` &\_\_n)
- `__istream_type` & `operator>>` (`unsigned int` &\_\_n)
- `__istream_type` & `operator>>` (`long` &\_\_n)
- `__istream_type` & `operator>>` (`unsigned long` &\_\_n)
- `__istream_type` & `operator>>` (`long long` &\_\_n)
- `__istream_type` & `operator>>` (`unsigned long long` &\_\_n)
- `__istream_type` & `operator>>` (`float` &\_\_f)
- `__istream_type` & `operator>>` (`double` &\_\_f)
- `__istream_type` & `operator>>` (`long double` &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type` `get` ()
- `__istream_type` & `get` (`char_type` &\_\_c)
- `__istream_type` & `get` (`char_type` \*\_\_s, `streamsize` \_\_n, `char_type` \_\_delim)
- `__istream_type` & `get` (`char_type` \*\_\_s, `streamsize` \_\_n)
- `__istream_type` & `get` (`__streambuf_type` &\_\_sb, `char_type` \_\_delim)

- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ()
  - int\_type [peek](#) ()
  - [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator void \\*](#) () const
  - bool [operator!](#) () const

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)

- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

#### Protected Types

- enum { **\_S\_local\_word\_size** }

#### Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) & \_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
[\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT & \_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)

#### Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.622.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_istream< _CharT, _Traits, _Alloc >
```

Controlling input for std::string.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 272 of file `sstream`.

## 4.622.2 Member Typedef Documentation

4.622.2.1 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type` [inherited]

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.622.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i)` [inherited]

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.622.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`

- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file `ios_base.h`.

#### 4.622.2.4 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file `ios_base.h`.

#### 4.622.2.5 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file `ios_base.h`.

## 4.622.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

## 4.622.3 Member Enumeration Documentation

## 4.622.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

## 4.622.4 Constructor &amp; Destructor Documentation

## 4.622.4.1 template&lt;typename \_CharT, typename \_Traits, typename \_Alloc&gt; std::basic\_istream&lt;\_CharT, \_Traits, \_Alloc&gt;::basic\_istream( ios\_base::openmode \_\_mode = ios\_base::in ) [inline], [explicit]

Default constructor starts with an empty string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::in` is automatically included in `__mode`.

Initializes `sb` using `__mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 308 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

## 4.622.4.2 template&lt;typename \_CharT, typename \_Traits, typename \_Alloc&gt; std::basic\_istream&lt;\_CharT, \_Traits, \_Alloc&gt;::basic\_istream( const \_\_string\_type &amp; \_\_str, ios\_base::openmode \_\_mode = ios\_base::in ) [inline], [explicit]

Starts with an existing string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::in` is automatically included in `mode`.

Initializes `sb` using `str` and `mode` in, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 326 of file `sstream`.

References `std::basic_ios< _CharT, _Traits >::init()`.

**4.622.4.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_istream< _CharT, _Traits, _Alloc >::~~basic_istream( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 337 of file `sstream`.

#### 4.622.5 Member Function Documentation

**4.622.5.1** `const locale& std::ios_base::M_getloc( ) const [inline],[inherited]`

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

**4.622.5.2** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad( ) const [inline],[inherited]`

Fast error checking.

##### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 205 of file `basic_ios.h`.

**4.622.5.3** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.622.5.4** template<typename \_CharT, typename \_Traits> basic\_ios< \_CharT, \_Traits > & basic\_ios< \_CharT, \_Traits >::copyfmt ( const basic\_ios< \_CharT, \_Traits > & \_\_rhs ) [inherited]

Copies fields of \_\_rhs into this.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

#### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

References std::basic\_ios< \_CharT, \_Traits >::exceptions(), std::basic\_ios< \_CharT, \_Traits >::fill(), std::ios\_base::flags(), std::ios\_base::getloc(), std::ios\_base::precision(), std::basic\_ios< \_CharT, \_Traits >::tie(), std::tie(), and std::ios\_base::width().

**4.622.5.5** template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic\_ios.h.

**4.622.5.6** template<typename \_CharT, typename \_Traits> iostate std::basic\_ios< \_CharT, \_Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt().

4.622.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except ) [inline], [inherited]`

Throwing exceptions on errors.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file `basic_ios.h`.

4.622.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Definition at line 195 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator void *()`, `std::basic_ios< char, _Traits >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

4.622.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 364 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

4.622.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )`  
`[inline], [inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ' ) in the current locale.

Definition at line 384 of file `basic_ios.h`.

4.622.5.11 `fmtflags std::ios_base::flags ( ) const` `[inline], [inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.622.5.12 `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline], [inherited]`

Setting new format flags all at once.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

#### Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

4.622.5.13 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( )`  
`const` `[inline], [inherited]`

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file `istream`.

4.622.5.14 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::int_type basic_istream< _CharT, _Traits>::get( void ) [inherited]`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits>::rdbuf(), and std::basic\_ios< \_CharT, \_Traits>::setstate().

4.622.5.15 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::get( char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

#### Returns

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns traits::eof().

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits>::rdbuf(), and std::basic\_ios< \_CharT, \_Traits>::setstate().

4.622.5.16 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::get( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.622.5.17** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline], [inherited]`

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s, __n, widen("\n"))`.

Definition at line 354 of file istream.

**4.622.5.18** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::sputc()`.

**4.622.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline],[inherited]`

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

**4.622.5.20** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.622.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline], [inherited]`

String extraction.

#### Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

#### Returns

`*this`

Returns `getline(__s, __n, widen('\n'))`.

Definition at line 427 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**4.622.5.22** `locale std::ios_base::getloc ( ) const [inline], [inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::ws()`.

**4.622.5.23** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.622.5.24** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim ) [inherited]`

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**4.622.5.25** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

4.622.5.26 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.622.5.27 `template<typename _CharT, typename _Traits > locale basic_ios< _CharT, _Traits >::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 114 of file basic\_ios.tcc.

References std::ios\_base::imbue().

Referenced by std::operator<<().

4.622.5.28 `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init ( basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_ios< char, \_Traits >::basic\_ios(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >::basic\_ostringstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

4.622.5.29 `long& std::ios_base::iword ( int __ix ) [inline], [inherited]`

Access to integer array.

## Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

## Returns

A reference to an integer associated with the index.

The `ixword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

**4.622.5.30** `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char __dfault ) const` `[inline]`, `[inherited]`

Squeezes characters.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file `basic_ios.h`.

**4.622.5.31** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const` `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**4.622.5.32** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const` `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file `basic_ios.h`.

4.622.5.33 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( __istream_type &(*)(__istream_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file `istream`.

4.622.5.34 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( __ios_type &(*)(__ios_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 124 of file `istream`.

4.622.5.35 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( ios_base &(*)(ios_base &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

4.622.5.36 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( bool & __n ) [inline], [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

4.622.5.37 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::operator>>( short & __n ) [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.622.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned short & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.622.5.39** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.622.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned int & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 182 of file istream.

**4.622.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( long & __n ) [inline],[inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 186 of file istream.

**4.622.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( unsigned long & __n ) [inline],[inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 190 of file istream.

**4.622.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( long long & __n ) [inline],[inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 195 of file istream.

4.622.5.44 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline],[inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

4.622.5.45 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( float & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

4.622.5.46 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( double & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

4.622.5.47 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::operator>> ( long double & __f ) [inline],[inherited]`

Floating point arithmetic extractors.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**4.622.5.48** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( void *& __p ) [inline], [inherited]`

Basic arithmetic extractors.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.622.5.49** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.622.5.50 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type basic_istream<_CharT, _Traits>::peek( void ) [inherited]`

Looking ahead in the stream.

#### Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.622.5.51 `streamsize std::ios_base::precision( ) const [inline], [inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

4.622.5.52 `streamsize std::ios_base::precision( streamsize __prec ) [inline], [inherited]`

Changing flags.

#### Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

#### Returns

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

4.622.5.53 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::putback( char_type __c ) [inherited]`

Unextracting a single character.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**4.622.5.54** `void*& std::ios_base::pword ( int __ix )` `[inline]`, `[inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file `ios_base.h`.

**4.622.5.55** `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf ( basic_streambuf< _CharT, _Traits > * __sb )` `[inherited]`

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a

result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.622.5.56** template<typename \_CharT, typename \_Traits, typename \_Alloc> \_\_stringbuf\_type\* std::basic\_istream<\_CharT, \_Traits, \_Alloc>::rdbuf( ) const [inline]

Accessing the underlying buffer.

#### Returns

The current basic\_stringbuf buffer.

This hides both signatures of std::basic\_ios::rdbuf().

Definition at line 348 of file sstream.

**4.622.5.57** template<typename \_CharT, typename \_Traits> iostate std::basic\_ios<\_CharT, \_Traits>::rdstate( ) const [inline], [inherited]

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See std::ios\_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::bad(), std::basic\_ios<char, \_Traits>::eof(), std::basic\_ios<char, \_Traits>::fail(), std::basic\_ios<char, \_Traits>::good(), std::basic\_istream<\_CharT, \_Traits>::putback(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ios<char, \_Traits>::setstate(), and std::basic\_istream<\_CharT, \_Traits>::unset().

**4.622.5.58** template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & basic\_istream<\_CharT, \_Traits>::read( char\_type \* \_\_s, streamsize \_\_n ) [inherited]

Extraction without delimiters.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

#### Returns

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.622.5.59** template<typename \_CharT, typename \_Traits > streamsize basic\_istream< \_CharT, \_Traits >::readsome (char\_type \* \_\_s, streamsize \_\_n) [inherited]

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called A here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::min(), std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.622.5.60** void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.622.5.61** template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::seekg ( pos\_type \_\_pos ) [inherited]

Changing the current read position.

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets `failbit`.

## Note

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.622.5.62** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::seekg( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

## Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets `failbit`.

## Note

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.622.5.63** `fmtflags std::ios_base::setf( fmtflags __fmtfl ) [inline], [inherited]`

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**4.622.5.64** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline], [inherited]`

Setting new format flags.

**Parameters**

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios\_base.h.

**4.622.5.65** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline], [inherited]`

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See std::ios\_base::iostate for the possible bit values.

Definition at line 151 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::M\_write(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

4.622.5.66 `template<typename _CharT, typename _Traits, typename _Alloc> __string_type std::basic_istream< _CharT, _Traits, _Alloc>::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

`rdbuf ()->str ()`

Definition at line 356 of file sstream.

4.622.5.67 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_istream< _CharT, _Traits, _Alloc>::str ( const __string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf ()->str (s)`.

Definition at line 366 of file sstream.

4.622.5.68 `template<typename _CharT, typename _Traits> int basic_istream< _CharT, _Traits>::sync ( void ) [inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf ()` is a null pointer, returns -1.

Otherwise, calls `rdbuf ()->pubsync ()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount ()`.

Definition at line 781 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits>::pubsync()`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.622.5.69 `static bool std::ios_base::sync_with_stdio ( bool __sync = true ) [static], [inherited]`

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.622.5.70** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg ( void )` [inherited]

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**4.622.5.71** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const` [inline], [inherited]

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.622.5.72** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( basic_ostream< _CharT, _Traits >* __tiestr )` [inline], [inherited]

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic\_ios.h.

**4.622.5.73** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::unget ( void ) [inherited]`

Unextracting the previous character.

**Returns**

\*this

If rdbuf() is not null, calls rdbuf()->sungetc(c).

If rdbuf() is null or if sungetc() fails, sets badbit in the error state.

**Note**

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits>::clear(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits>::rdbuf(), std::basic\_ios< \_CharT, \_Traits>::rdstate(), std::basic\_ios< \_CharT, \_Traits>::setstate(), and std::basic\_streambuf< \_CharT, \_Traits>::sungetc().

**4.622.5.74** `void std::ios_base::unsetf ( fmtflags __mask ) [inline], [inherited]`

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**4.622.5.75** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits>::widen ( char __c ) const [inline], [inherited]`

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios< char, _Traits >::fill()`, `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::getline()`, `std::tr2::operator>>()`, and `std::operator>>()`.

**4.622.5.76** `streamsize std::ios_base::width ( ) const` `[inline]`, `[inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**4.622.5.77** `streamsize std::ios_base::width ( streamsize __wide )` `[inline]`, `[inherited]`

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

**4.622.5.78** `static int std::ios_base::xalloc ( ) throw ()` `[static]`, `[inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

#### 4.622.6 Member Data Documentation

**4.622.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount`  
`[protected], [inherited]`

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

**4.622.6.2** `const fmtflags std::ios_base::adjustfield` `[static], [inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.622.6.3** `const openmode std::ios_base::app` `[static], [inherited]`

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.622.6.4** `const openmode std::ios_base::ate` `[static], [inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**4.622.6.5** `const iostate std::ios_base::badbit` `[static], [inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

**4.622.6.6** `const fmtflags std::ios_base::basefield` `[static], [inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**4.622.6.7** `const seekdir std::ios_base::beg` `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**4.622.6.8** `const openmode std::ios_base::binary` `[static]`, `[inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**4.622.6.9** `const fmtflags std::ios_base::boolalpha` `[static]`, `[inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**4.622.6.10** `const seekdir std::ios_base::cur` `[static]`, `[inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**4.622.6.11** `const fmtflags std::ios_base::dec` `[static]`, `[inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

**4.622.6.12** `const seekdir std::ios_base::end` `[static]`, `[inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**4.622.6.13** `const iostate std::ios_base::eofbit` `[static]`, `[inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsomewhat(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 4.622.6.14 const iostate std::ios\_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.622.6.15 const fmtflags std::ios\_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.622.6.16 const fmtflags std::ios\_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.622.6.17 const iostate std::ios\_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsomewhat(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.622.6.18 const fmtflags std::ios\_base::hex** [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.622.6.19 const openmode std::ios\_base::in** [static], [inherited]

Open for input. Default for ifstream and fstream.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**4.622.6.20 const fmtflags std::ios\_base::internal** [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

**4.622.6.21 const fmtflags std::ios\_base::left** [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.622.6.22 const fmtflags std::ios\_base::oct** [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.622.6.23 const openmode std::ios\_base::out** [static], [inherited]

Open for output. Default for ofstream and fstream.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**4.622.6.24 const fmtflags std::ios\_base::right** [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.622.6.25** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.622.6.26** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.622.6.27** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.622.6.28** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.622.6.29** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::skipws()`.

**4.622.6.30** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.622.6.31** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, `std::unitbuf()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**4.622.6.32** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

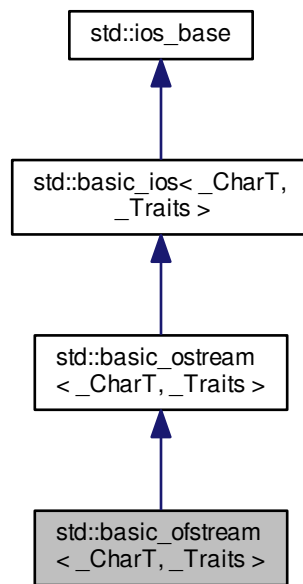
Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [sstream](#)

#### 4.623 std::basic\_ofstream< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_ofstream< \_CharT, \_Traits >:



#### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event __e, ios_base & __b, int __i)`

- typedef \_ios\_Fmtflags [fmtflags](#)
- typedef traits\_type::int\_type **int\_type**
- typedef int **io\_state**
- typedef \_ios\_istate [iostate](#)
- typedef traits\_type::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_ios\_Openmode [openmode](#)
- typedef traits\_type::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_ios\_Seekdir [seekdir](#)
- typedef [std::streamoff](#) **streamoff**
- typedef [std::streampos](#) **streampos**
- typedef \_Traits **traits\_type**
- typedef [num\\_get](#)< \_CharT,  
[istreambuf\\_iterator](#)< \_CharT,  
\_Traits > > [\\_\\_num\\_get\\_type](#)

#### Public Member Functions

- [basic\\_ofstream](#) ()
- [basic\\_ofstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out|ios\\_base::trunc](#))
- [basic\\_ofstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out|ios\\_base::trunc](#))
- [~basic\\_ofstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
- bool [is\\_open](#) ()
- bool [is\\_open](#) () const
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- void [open](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out|ios\\_base::trunc](#))
- void [open](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out|ios\\_base::trunc](#))
- [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \* \_\_p)
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)

- [streamsize precision](#) () const
- [streamsize precision](#) (streamsize \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- [\\_\\_filebuf\\_type](#) \* [rdbuf](#) () const
- [iostate rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [\\_\\_ostream\\_type](#) & [seekp](#) (pos\_type)
- [\\_\\_ostream\\_type](#) & [seekp](#) (off\_type, [ios\\_base::seekdir](#))
- [fmtflags self](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags self](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([iostate](#) \_\_state)
- pos\_type [tellp](#) ()
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- char\_type [widen](#) (char \_\_c) const
- [streamsize width](#) () const
- [streamsize width](#) (streamsize \_\_wide)
- [\\_\\_ostream\\_type](#) & [operator<<](#) ( [\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ( [\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [operator void \\*](#) () const
- bool [operator!](#) () const

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

#### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
[\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.623.1 Detailed Description

```
template<typename _CharT, typename _Traits>class std::basic_ofstream< _CharT, _Traits >
```

Controlling output for files.

## Template Parameters

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                          |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> . |

This class supports reading from named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as [sb](#).

Definition at line 599 of file [fstream](#).

## 4.623.2 Member Typedef Documentation

4.623.2.1 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]`

These are non-standard types.

Definition at line 90 of file `basic_ios.h`.

4.623.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.623.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.623.2.4 typedef \_Ios\_Iostate std::ios\_base::iostate [inherited]

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file *ios\_base.h*.

#### 4.623.2.5 typedef \_Ios\_Openmode std::ios\_base::openmode [inherited]

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file *ios\_base.h*.

#### 4.623.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type *seekdir* are:

- beg
- cur, equivalent to *SEEK\_CUR* in the C standard library.
- end, equivalent to *SEEK\_END* in the C standard library.

Definition at line 393 of file *ios\_base.h*.

### 4.623.3 Member Enumeration Documentation

#### 4.623.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

*erase\_event* is used during *~ios()* and *copyfmt()*. *imbue\_event* is used during *imbue()*. *copyfmt\_event* is used during *copyfmt()*.

Definition at line 419 of file *ios\_base.h*.

## 4.623.4 Constructor &amp; Destructor Documentation

4.623.4.1 `template<typename _CharT, typename _Traits> std::basic_ofstream< _CharT, _Traits >::basic_ofstream ( )`  
`[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 625 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::init()`.

4.623.4.2 `template<typename _CharT, typename _Traits> std::basic_ofstream< _CharT, _Traits >::basic_ofstream ( const char * __s, ios_base::openmode __mode = ios_base::out|ios_base::trunc )` `[inline],[explicit]`

Create an output file stream.

## Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::out` | `ios_base::trunc` is automatically included in `__mode`.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 640 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::init()`, and `std::basic_ofstream< _CharT, _Traits >::open()`.

4.623.4.3 `template<typename _CharT, typename _Traits> std::basic_ofstream< _CharT, _Traits >::basic_ofstream ( const std::string & __s, ios_base::openmode __mode = ios_base::out|ios_base::trunc )` `[inline],[explicit]`

Create an output file stream.

## Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | <code>std::string</code> specifying the filename.              |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::out` | `ios_base::trunc` is automatically included in `__mode`.

Definition at line 658 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::init()`, and `std::basic_ofstream< _CharT, _Traits >::open()`.

4.623.4.4 `template<typename _CharT, typename _Traits> std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream ( )`  
`[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 673 of file `fstream`.

## 4.623.5 Member Function Documentation

4.623.5.1 **const locale& std::ios\_base::M.getloc ( ) const** [inline],[inherited]

Locale access.

#### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::time\_put< \_CharT, \_Outlter >::put().

4.623.5.2 **template<typename \_CharT, typename \_Traits> void std::basic\_ostream< \_CharT, \_Traits >::M.write ( const char\_type \* \_\_s, streamsize \_\_n )** [inline],[inherited]

Core write functionality, without sentry.

#### Parameters

|            |                                         |
|------------|-----------------------------------------|
| <b>__s</b> | The array to insert.                    |
| <b>__n</b> | Maximum number of characters to insert. |

Definition at line 311 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

4.623.5.3 **template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::bad ( ) const** [inline],[inherited]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

4.623.5.4 **template<typename \_CharT, typename \_Traits> void basic\_ios< \_CharT, \_Traits >::clear ( iostate \_\_state = goodbit )** [inherited]

[Re]sets the error state.

#### Parameters

|                |                               |
|----------------|-------------------------------|
| <b>__state</b> | The new state flag(s) to set. |
|----------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**4.623.5.5** `template<typename _CharT, typename _Traits> void std::basic_ofstream< _CharT, _Traits >::close ( )`  
`[inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 753 of file `fstream`.

References `std::basic_filebuf< _CharT, _Traits >::close()`, `std::ios_base::failbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.623.5.6** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs )` `[inherited]`

Copies fields of `__rhs` into this.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, `std::tie()`, and `std::ios_base::width()`.

**4.623.5.7** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const` `[inline]`,  
`[inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file `basic_ios.h`.

**4.623.5.8** `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const`  
`[inline]`, `[inherited]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt().

**4.623.5.9** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except ) [inline],[inherited]`

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

**4.623.5.10** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline],[inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**4.623.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const`  
`[inline], [inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 364 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

**4.623.5.12** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )`  
`[inline], [inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file `basic_ios.h`.

**4.623.5.13** `fmtflags std::ios_base::flags ( ) const` `[inline], [inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.623.5.14** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline], [inherited]`

Setting new format flags all at once.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

**4.623.5.15** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::flush ( ) [inherited]`

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( )->pubsync ( )`, and if that returns `-1`, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**4.623.5.16** `locale std::ios_base::getloc ( ) const [inline],[inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue (loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale ( )`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`, `std::money_put< _CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::ws()`.

**4.623.5.17** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits>::good ( ) const [inline],[inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits>::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits>::sentry::sentry()`.

4.623.5.18 `template<typename _CharT, typename _Traits > locale basic_ios< _CharT, _Traits >::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 114 of file `basic_ios.tcc`.

References `std::ios_base::imbue()`.

Referenced by `std::operator<<()`.

4.623.5.19 `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init ( basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >::basic_ostreamstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.

4.623.5.20 `template<typename _CharT, typename _Traits > bool std::basic_ofstream< _CharT, _Traits >::is_open ( ) [inline]`

Wrapper to test for an open file.

#### Returns

`rdbuf()->is_open()`

Definition at line 692 of file `fstream`.

References `std::basic_filebuf< _CharT, _Traits >::is_open()`.

4.623.5.21 `long& std::ios_base::iword ( int __ix ) [inline], [inherited]`

Access to integer array.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `isword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

**4.623.5.22** `template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow ( char_type __c, char __dfault ) const` `[inline]`, `[inherited]`

Squeezes characters.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file `basic_ios.h`.

**4.623.5.23** `template<typename _CharT, typename _Traits > void std::basic_ofstream< _CharT, _Traits >::open ( const char * __s, ios_base::openmode __mode = ios_base::out | ios_base::trunc )` `[inline]`

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 713 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::failbit`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`.

4.623.5.24 `template<typename _CharT, typename _Traits> void std::basic_ofstream< _CharT, _Traits >::open ( const std::string& __s, ios_base::openmode __mode = ios_base::out | ios_base::trunc ) [inline]`

Opens an external file.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 734 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::failbit`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.623.5.25 `template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::operator void * ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

4.623.5.26 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file `basic_ios.h`.

4.623.5.27 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

4.623.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ios_type &(*)(__ios_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

4.623.5.29 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( ios_base &(*)(ios_base &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For

more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

**4.623.5.30** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<<( long __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

**4.623.5.31** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<<( unsigned long __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**4.623.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<<( bool __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 174 of file ostream.

**4.623.5.33** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( short __n ) [inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios\_base::basefield, std::ios\_base::flags(), std::ios\_base::hex, and std::ios\_base::oct.

**4.623.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( unsigned short __n ) [inline],[inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 181 of file ostream.

**4.623.5.35** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( int __n ) [inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios\_base::basefield, std::ios\_base::flags(), std::ios\_base::hex, and std::ios\_base::oct.

**4.623.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<<( unsigned int __n ) [inline],[inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

**4.623.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<<( long long __n ) [inline],[inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

**4.623.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<<( unsigned long long __n ) [inline],[inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 205 of file ostream.

**4.623.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( double __f ) [inline],[inherited]`

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 220 of file ostream.

**4.623.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( float __f ) [inline],[inherited]`

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**4.623.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( long double __f ) [inline],[inherited]`

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.623.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< ( const void * __p ) [inline], [inherited]`

Pointer arithmetic inserters.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

4.623.5.43 `template<typename _CharT, typename _Traits > basic_ofstream< _CharT, _Traits > & basic_ofstream< _CharT, _Traits >::operator<< ( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.623.5.44 `streamsize std::ios_base::precision ( ) const [inline], [inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

4.623.5.45 **streamsize** std::ios\_base::precision ( **streamsize** \_\_prec ) [inline],[inherited]

Changing flags.

#### Parameters

|        |                          |
|--------|--------------------------|
| __prec | The new precision value. |
|--------|--------------------------|

#### Returns

The previous value of precision().

Definition at line 630 of file ios\_base.h.

4.623.5.46 **template**<typename \_CharT, typename \_Traits > **basic\_ostream**< \_CharT, \_Traits > & **basic\_ostream**< \_CharT, \_Traits >::put ( **char\_type** \_\_c ) [inherited]

Simple insertion.

#### Parameters

|     |                          |
|-----|--------------------------|
| __c | The character to insert. |
|-----|--------------------------|

#### Returns

\*this

Tries to insert \_\_c.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.623.5.47 **void\***& std::ios\_base::pword ( **int** \_\_ix ) [inline],[inherited]

Access to void pointer array.

#### Parameters

|      |                       |
|------|-----------------------|
| __ix | Index into the array. |
|------|-----------------------|

#### Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

4.623.5.48 `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf ( basic_streambuf< _CharT, _Traits > * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

4.623.5.49 `template<typename _CharT, typename _Traits > __filebuf_type* std::basic_ofstream< _CharT, _Traits >::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 684 of file fstream.

4.623.5.50 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file basic\_ios.h.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

4.623.5.51 void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]

Add the callback \_\_fn with parameter \_\_index.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.623.5.52 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::seekp ( pos\_type \_\_pos ) [inherited]

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

\*this

If fail() is not true, calls rdbuf()->pubseekpos(pos). If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::out, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.623.5.53 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::seekp ( off\_type \_\_off, ios\_base::seekdir \_\_dir ) [inherited]

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

\*this

If fail() is not true, calls rdbuf()->pubseekoff(off, dir). If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::out, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.623.5.54 fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl ) [inline], [inherited]

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**4.623.5.55** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline], [inherited]`

Setting new format flags.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

## Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios\_base.h.

**4.623.5.56** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline], [inherited]`

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See std::ios\_base::iostate for the possible bit values.

Definition at line 151 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::M\_write(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unset(), and std::ws().

4.623.5.57 static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true ) [static],[inherited]

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

4.623.5.58 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits >::pos\_type basic\_ostream< \_CharT, \_Traits >::tellp ( ) [inherited]

Getting the current write position.

#### Returns

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf() ->pubseekoff(0, cur, out).

Definition at line 237 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::cur, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::out, and std::basic\_ios< \_CharT, \_Traits >::rdbuf().

4.623.5.59 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie ( ) const [inline],[inherited]

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, std::cin is tied to std::cout.

Definition at line 289 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

4.623.5.60 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie ( basic\_ostream< \_CharT, \_Traits > \* \_\_tiestr ) [inline],[inherited]

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic\_ios.h.

**4.623.5.61** void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline],[inherited]

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**4.623.5.62** template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios< \_CharT, \_Traits >::widen ( char \_\_c )  
const [inline],[inherited]

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios< char, \_Traits >::fill(), std::basic\_istream< char >::get(), std::basic\_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

**4.623.5.63** streamsize std::ios\_base::width ( ) const [inline],[inherited]

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_Outiter >::do\_put(), and std::operator>>().

**4.623.5.64** streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline],[inherited]

Changing flags.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

#### Returns

The previous value of width().

Definition at line 653 of file ios\_base.h.

**4.623.5.65** template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::write ( const char\_type \* \_\_s, streamsize \_\_n ) [inherited]

Character string insertion.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

#### Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic\_ostream< \_CharT, \_Traits >::\_M\_write(), and std::ios\_base::badbit.

**4.623.5.66** static int std::ios\_base::xalloc ( ) throw () [static],[inherited]

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.623.6 Member Data Documentation****4.623.6.1 const fmtflags std::ios\_base::adjustfield** [static],[inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.623.6.2 const openmode std::ios\_base::app** [static],[inherited]

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.623.6.3 const openmode std::ios\_base::ate** [static],[inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**4.623.6.4 const iostate std::ios\_base::badbit** [static],[inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

**4.623.6.5 const fmtflags std::ios\_base::basefield** [static],[inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**4.623.6.6 const seekdir std::ios\_base::beg** [static],[inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**4.623.6.7 const openmode std::ios\_base::binary** [static],[inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**4.623.6.8 const fmtflags std::ios\_base::boolalpha** [static],[inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

**4.623.6.9 const seekdir std::ios\_base::cur** [static],[inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**4.623.6.10 const fmtflags std::ios\_base::dec** [static],[inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

**4.623.6.11 const seekdir std::ios\_base::end** [static],[inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**4.623.6.12 const iostate std::ios\_base::eofbit** [static],[inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits

>::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 4.623.6.13 const iostate std::ios\_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.623.6.14 const fmtflags std::ios\_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.623.6.15 const fmtflags std::ios\_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.623.6.16 const iostate std::ios\_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 4.623.6.17 const fmtflags std::ios\_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.623.6.18** `const openmode std::ios_base::in` `[static], [inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**4.623.6.19** `const fmtflags std::ios_base::internal` `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

**4.623.6.20** `const fmtflags std::ios_base::left` `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.623.6.21** `const fmtflags std::ios_base::oct` `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.623.6.22** `const openmode std::ios_base::out` `[static], [inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**4.623.6.23** `const fmtflags std::ios_base::right` `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios\_base.h.

Referenced by std::right().

**4.623.6.24** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.623.6.25** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.623.6.26** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.623.6.27** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.623.6.28** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::skipws()`.

**4.623.6.29** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.623.6.30** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, `std::unitbuf()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**4.623.6.31** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [fstream](#)



- typedef num\_get< \_CharT,  
istreambuf\_iterator< \_CharT,  
\_Traits > > \_\_num\_get\_type

#### Public Member Functions

- basic\_ostream ( \_\_streambuf\_type \* \_\_sb)
- virtual ~basic\_ostream ()
- const locale & \_M\_getloc () const
- void \_M\_setstate (iostate \_\_state)
- bool bad () const
- void clear (iostate \_\_state=goodbit)
- basic\_ios & copyfmt (const basic\_ios & \_\_rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate \_\_except)
- bool fail () const
- char\_type fill () const
- char\_type fill (char\_type \_\_ch)
- fmtflags flags () const
- fmtflags flags (fmtflags \_\_fmtfl)
- \_\_ostream\_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale & \_\_loc)
- long & iword (int \_\_ix)
- char narrow (char\_type \_\_c, char \_\_dfault) const
- \_\_ostream\_type & operator<< (const void \* \_\_p)
- \_\_ostream\_type & operator<< ( \_\_streambuf\_type \* \_\_sb)
- streamsize precision () const
- streamsize precision (streamsize \_\_prec)
- void \*& pword (int \_\_ix)
- basic\_streambuf< \_CharT,  
\_Traits > \* rdbuf () const
- basic\_streambuf< \_CharT,  
\_Traits > \* rdbuf (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
- iostate rdstate () const
- void register\_callback (event\_callback \_\_fn, int \_\_index)
- \_\_ostream\_type & seekp (pos\_type)
- \_\_ostream\_type & seekp (off\_type, ios\_base::seekdir)
- fmtflags setf (fmtflags \_\_fmtfl)
- fmtflags setf (fmtflags \_\_fmtfl, fmtflags \_\_mask)
- void setstate (iostate \_\_state)
- pos\_type tellp ()
- basic\_ostream< \_CharT, \_Traits > \* tie () const
- basic\_ostream< \_CharT, \_Traits > \* tie (basic\_ostream< \_CharT, \_Traits > \* \_\_tiesr)
- void unsetf (fmtflags \_\_mask)
- char\_type widen (char \_\_c) const
- streamsize width () const
- streamsize width (streamsize \_\_wide)

- [\\_\\_ostream\\_type & operator<< \(\\_\\_ostream\\_type &\(\\_\\_pf\)\(\\_\\_ostream\\_type &\)\)](#)
- [\\_\\_ostream\\_type & operator<< \(\\_\\_ios\\_type &\(\\_\\_pf\)\(\\_\\_ios\\_type &\)\)](#)
- [\\_\\_ostream\\_type & operator<< \(ios\\_base &\(\\_\\_pf\)\(ios\\_base &\)\)](#)

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type & operator<< \(long \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(unsigned long \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(bool \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(short \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(unsigned short \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(int \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(unsigned int \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(long long \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & operator<< \(unsigned long long \\_\\_n\)](#)
- 
- [\\_\\_ostream\\_type & operator<< \(double \\_\\_f\)](#)
  - [\\_\\_ostream\\_type & operator<< \(float \\_\\_f\)](#)
  - [\\_\\_ostream\\_type & operator<< \(long double \\_\\_f\)](#)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type & put \(char\\_type \\_\\_c\)](#)
  - [void \\_M\\_write \(const char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
  - [\\_\\_ostream\\_type & write \(const char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
- 
- [operator void \\* \(\) const](#)
  - [bool operator! \(\) const](#)

### Static Public Member Functions

- [static bool sync\\_with\\_stdio \(bool \\_\\_sync=true\)](#)
- [static int xalloc \(\) throw \(\)](#)

## Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename `_ValueT` >  
    `__ostream_type` & `_M_insert` (`_ValueT` \_\_v)
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

## Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- `const` `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- `bool` `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- `const` `__num_get_type` \* `_M_num_get`
- `const` `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- `int` `_M_word_size`
- `_Words` `_M_word_zero`

## Friends

- class `sentry`

## 4.624.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ostream< _CharT, _Traits >
```

Template class `basic_ostream`.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

Definition at line 58 of file `ostream`.

## 4.624.2 Member Typedef Documentation

```
4.624.2.1 template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> >
std::basic_ios< _CharT, _Traits >::__num_get_type [inherited]
```

These are non-standard types.

Definition at line 90 of file `basic_ios.h`.

4.624.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i)` [inherited]

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.624.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.624.2.4 typedef \_Ios\_Iostate std::ios\_base::iostate [inherited]

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios\_base.h.

#### 4.624.2.5 typedef \_Ios\_Openmode std::ios\_base::openmode [inherited]

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios\_base.h.

#### 4.624.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios\_base.h.

### 4.624.3 Member Enumeration Documentation

#### 4.624.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file ios\_base.h.

## 4.624.4 Constructor &amp; Destructor Documentation

4.624.4.1 `template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::basic_ostream ( __streambuf_type * __sb ) [inline],[explicit]`

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 84 of file ostream.

4.624.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream ( ) [inline],[virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 93 of file ostream.

## 4.624.5 Member Function Documentation

4.624.5.1 `const locale& std::ios_base::M_getloc ( ) const [inline],[inherited]`

Locale access.

## Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, and `std::time_put< _CharT, _Outlter >::put()`.

4.624.5.2 `template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::M_write ( const char_type * __s, streamsize __n ) [inline]`

Core write functionality, without sentry.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 311 of file ostream.

Referenced by `std::basic_ostream< _CharT, _Traits >::write()`.

4.624.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline],[inherited]`

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

**4.624.5.4** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.624.5.5** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of \_\_rhs into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

References std::basic\_ios< \_CharT, \_Traits >::exceptions(), std::basic\_ios< \_CharT, \_Traits >::fill(), std::ios\_base::flags(), std::ios\_base::getloc(), std::ios\_base::precision(), std::basic\_ios< \_CharT, \_Traits >::tie(), std::tie(), and std::ios\_base::width().

**4.624.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic\_ios.h.

4.624.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const`  
`[inline], [inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

4.624.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except )`  
`[inline], [inherited]`

Throwing exceptions on errors.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file `basic_ios.h`.

4.624.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const`  
`[inline], [inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file `basic_ios.h`.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**4.624.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const`  
`[inline], [inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 364 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::basic\_ios< char, \_Traits >::fill().

**4.624.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )`  
`[inline], [inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ' ) in the current locale.

Definition at line 384 of file basic\_ios.h.

**4.624.5.12** `fmtflags std::ios_base::flags ( ) const` `[inline], [inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.624.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline], [inherited]`

Setting new format flags all at once.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

**4.624.5.14** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::flush ( )`

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns `-1`, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**4.624.5.15** `locale std::ios_base::getloc ( ) const [inline], [inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::ws()`.

**4.624.5.16** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**4.624.5.17** `template<typename _CharT, typename _Traits> locale basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 114 of file `basic_ios.tcc`.

References `std::ios_base::imbue()`.

Referenced by `std::operator<<()`.

**4.624.5.18** `template<typename _CharT, typename _Traits> void basic_ios<_CharT, _Traits>::init ( basic_streambuf<_CharT, _Traits> * __sb ) [protected], [inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`, `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`, `std::basic_ios<char, _Traits>::basic_ios()`, `std::basic_istream<char>::basic_istream()`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>::basic_istreamstream()`, `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`, `std::basic_ostream<char>::basic_ostream()`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>::basic_ostreamstream()`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream()`.

**4.624.5.19** `long& std::ios_base::iword ( int __ix ) [inline], [inherited]`

Access to integer array.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

4.624.5.20 `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char __dfault ) const [inline], [inherited]`

Squeezes characters.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file `basic_ios.h`.

4.624.5.21 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

4.624.5.22 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file `basic_ios.h`.

4.624.5.23 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

4.624.5.24 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type &(*)(__ios_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

4.624.5.25 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

4.624.5.26 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

4.624.5.27 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

4.624.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( bool __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.624.5.29 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::operator<< ( short __n )`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

4.624.5.30 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned short __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.624.5.31 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::operator<< ( int __n )`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

4.624.5.32 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file `ostream`.

4.624.5.33 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long long __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

4.624.5.34 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long long __n ) [inline]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

4.624.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( double __f ) [inline]`

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

**4.624.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( float __f ) [inline]`

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**4.624.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long double __f ) [inline]`

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**4.624.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline]`

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.624.5.39** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::operator<<( __streambuf_type * __sb )`

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.624.5.40** `streamsize std::ios_base::precision( ) const [inline], [inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**4.624.5.41** `streamsize std::ios_base::precision( streamsize __prec ) [inline], [inherited]`

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().

Definition at line 630 of file ios\_base.h.

4.624.5.42 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::put ( char_type __c )`

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

4.624.5.43 `void*& std::ios_base::pword ( int __ix )` `[inline]`, `[inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

4.624.5.44 `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits>*& std::basic_ios< _CharT, _Traits >::rdbuf ( ) const` `[inline]`, `[inherited]`

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.624.5.45** template<typename \_CharT, typename \_Traits> **basic\_streambuf**< \_CharT, \_Traits > \* **basic\_ios**< \_CharT, \_Traits >::rdbuf ( **basic\_streambuf**< \_CharT, \_Traits > \* \_\_sb ) [inherited]

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.624.5.46** template<typename \_CharT, typename \_Traits> **iosstate** std::basic\_ios< \_CharT, \_Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See std::ios\_base::iosstate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

4.624.5.47 void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]

Add the callback \_\_fn with parameter \_\_index.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.624.5.48 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::seekp ( pos\_type \_\_pos )

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

\*this

If fail() is not true, calls rdbuf()->pubseekpos(pos). If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::out, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.624.5.49 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::seekp ( off\_type \_\_off, ios\_base::seekdir \_\_dir )

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

\*this

If fail() is not true, calls rdbuf()->pubseekoff(off, dir). If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::out, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.624.5.50 fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl ) [inline], [inherited]

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios\_base.h.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.624.5.51** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

## Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios\_base.h.

**4.624.5.52** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline]`, `[inherited]`

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file basic\_ios.h.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::tr2::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unset()`, and `std::ws()`.

4.624.5.53 static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true ) [static],[inherited]

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

4.624.5.54 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits >::pos\_type basic\_ostream< \_CharT, \_Traits >::tellp ( )

Getting the current write position.

#### Returns

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf() ->pubseekoff(0, cur, out).

Definition at line 237 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::cur, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::out, and std::basic\_ios< \_CharT, \_Traits >::rdbuf().

4.624.5.55 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie ( ) const [inline],[inherited]

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, std::cin is tied to std::cout.

Definition at line 289 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

4.624.5.56 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie ( basic\_ostream< \_CharT, \_Traits > \* \_\_tiestr ) [inline],[inherited]

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic\_ios.h.

**4.624.5.57** void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline],[inherited]

Clearing format flags.

**Parameters**

|        |                     |
|--------|---------------------|
| __mask | The flags to unset. |
|--------|---------------------|

This function clears \_\_mask in the format flags.

Definition at line 610 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**4.624.5.58** template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios<\_CharT, \_Traits>::widen ( char \_\_c ) const [inline],[inherited]

Widens characters.

**Parameters**

|     |                         |
|-----|-------------------------|
| __c | The character to widen. |
|-----|-------------------------|

**Returns**

The widened character.

Maps a character of char to a character of char\_type.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios<char, \_Traits>::fill(), std::basic\_istream<char>::get(), std::basic\_istream<char>::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

**4.624.5.59** streamsize std::ios\_base::width ( ) const [inline],[inherited]

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_Outiter >::do\_put(), and std::operator>>().

**4.624.5.60** streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline],[inherited]

Changing flags.

#### Parameters

|        |                      |
|--------|----------------------|
| __wide | The new width value. |
|--------|----------------------|

#### Returns

The previous value of width().

Definition at line 653 of file ios\_base.h.

**4.624.5.61** template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::write ( const char\_type \* \_\_s, streamsize \_\_n )

Character string insertion.

#### Parameters

|     |                                         |
|-----|-----------------------------------------|
| __s | The array to insert.                    |
| __n | Maximum number of characters to insert. |

#### Returns

\*this

Characters are copied from \_\_s and inserted into the stream until one of the following happens:

- \_\_n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic\_ostream< \_CharT, \_Traits >::\_M\_write(), and std::ios\_base::badbit.

**4.624.5.62** static int std::ios\_base::xalloc ( ) throw () [static],[inherited]

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.624.6 Member Data Documentation****4.624.6.1 const fmtflags std::ios\_base::adjustfield** [static],[inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.624.6.2 const openmode std::ios\_base::app** [static],[inherited]

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.624.6.3 const openmode std::ios\_base::ate** [static],[inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**4.624.6.4 const iostate std::ios\_base::badbit** [static],[inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

**4.624.6.5 const fmtflags std::ios\_base::basefield** [static],[inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**4.624.6.6 const seekdir std::ios\_base::beg** [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**4.624.6.7 const openmode std::ios\_base::binary** [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**4.624.6.8 const fmtflags std::ios\_base::boolalpha** [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

**4.624.6.9 const seekdir std::ios\_base::cur** [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**4.624.6.10 const fmtflags std::ios\_base::dec** [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

**4.624.6.11 const seekdir std::ios\_base::end** [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**4.624.6.12 const iostate std::ios\_base::eofbit** [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits

>::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 4.624.6.13 const iostate std::ios\_base::failbit [static],[inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 4.624.6.14 const fmtflags std::ios\_base::fixed [static],[inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

#### 4.624.6.15 const fmtflags std::ios\_base::floatfield [static],[inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 4.624.6.16 const iostate std::ios\_base::goodbit [static],[inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 4.624.6.17 const fmtflags std::ios\_base::hex [static],[inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.624.6.18** `const openmode std::ios_base::in` `[static], [inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**4.624.6.19** `const fmtflags std::ios_base::internal` `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

**4.624.6.20** `const fmtflags std::ios_base::left` `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.624.6.21** `const fmtflags std::ios_base::oct` `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.624.6.22** `const openmode std::ios_base::out` `[static], [inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**4.624.6.23** `const fmtflags std::ios_base::right` `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios\_base.h.

Referenced by std::right().

**4.624.6.24** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.624.6.25** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.624.6.26** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.624.6.27** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.624.6.28** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

**4.624.6.29** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.624.6.30** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

**4.624.6.31** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [ostream](#)

- [ostream.tcc](#)

## 4.625 std::basic\_ostream< \_CharT, \_Traits >::sentry Class Reference

### Public Member Functions

- [sentry](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)
- [~sentry](#) ()
- [operator bool](#) () const

### 4.625.1 Detailed Description

template<typename \_CharT, typename \_Traits>class std::basic\_ostream< \_CharT, \_Traits >::sentry

Performs setup work for output streams.

Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 400 of file ostream.

### 4.625.2 Constructor & Destructor Documentation

4.625.2.1 template<typename \_CharT, typename \_Traits> std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry (   
 basic\_ostream< \_CharT, \_Traits > &\_\_os ) [explicit]

The constructor performs preparatory work.

#### Parameters

|      |                             |
|------|-----------------------------|
| __os | The output stream to guard. |
|------|-----------------------------|

If the stream state is good (\_\_os.good() is true), then if the stream is tied to another output stream, is.-tie()->flush() is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file ostream.tcc.

References std::ios\_base::failbit, std::basic\_ios< \_CharT, \_Traits >::good(), std::basic\_ios< \_CharT, \_Traits >::setstate(), and std::basic\_ios< \_CharT, \_Traits >::tie().

4.625.2.2 template<typename \_CharT, typename \_Traits> std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry ( )   
 [inline]

Possibly flushes the stream.

If ios\_base::unitbuf is set in os.flags(), and std::uncaught\_exception() is true, the sentry destructor calls flush() on the output stream.

Definition at line 428 of file ostream.

References std::ios\_base::badbit, std::uncaught\_exception(), and std::ios\_base::unitbuf.

### 4.625.3 Member Function Documentation

4.625.3.1 `template<typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>::sentry::operator bool ( )  
const [inline], [explicit]`

Quick status checking.

#### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

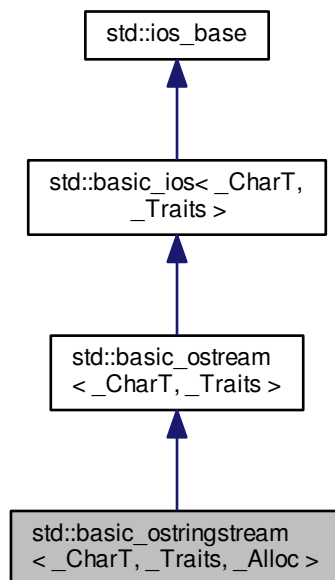
Definition at line 449 of file ostream.

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

## 4.626 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference

Inheritance diagram for std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>:



#### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`

- typedef num\_put<\_CharT, ostreambuf\_iterator<\_CharT, \_Traits>> \_\_num\_put\_type
- typedef basic\_ostream<char\_type, traits\_type> \_\_ostream\_type
- typedef basic\_streambuf<\_CharT, \_Traits> \_\_streambuf\_type
- typedef basic\_string<\_CharT, \_Traits, \_Alloc> \_\_string\_type
- typedef basic\_stringbuf<\_CharT, \_Traits, \_Alloc> \_\_stringbuf\_type
- typedef \_Alloc allocator\_type
- typedef \_CharT char\_type
- enum event { erase\_event, imbue\_event, copyfmt\_event }
- typedef void(\* event\_callback)(event \_\_e, ios\_base &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags fmtflags
- typedef traits\_type::int\_type int\_type
- typedef int io\_state
- typedef \_ios\_istate iostate
- typedef traits\_type::off\_type off\_type
- typedef int open\_mode
- typedef \_ios\_Openmode openmode
- typedef traits\_type::pos\_type pos\_type
- typedef int seek\_dir
- typedef \_ios\_Seekdir seekdir
- typedef std::streamoff streamoff
- typedef std::streampos streampos
- typedef \_Traits traits\_type
  
- typedef num\_get<\_CharT, istreambuf\_iterator<\_CharT, \_Traits>> \_\_num\_get\_type

#### Public Member Functions

- basic\_ostringstream (ios\_base::openmode \_\_mode=ios\_base::out)
- basic\_ostringstream (const \_\_string\_type &\_\_str, ios\_base::openmode \_\_mode=ios\_base::out)
- ~basic\_ostringstream ()
- const locale &\_M\_getloc () const
- void \_M\_setstate (iostate \_\_state)
- bool bad () const
- void clear (iostate \_\_state=goodbit)
- basic\_ios &copyfmt (const basic\_ios &\_\_rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate \_\_except)
- bool fail () const
- char\_type fill () const
- char\_type fill (char\_type \_\_ch)
- fmtflags flags () const
- fmtflags flags (fmtflags \_\_fmtfl)

- [\\_\\_ostream\\_type](#) & [flush](#) ()
  - [locale](#) [getloc](#) () const
  - bool [good](#) () const
  - [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
  - long & [iword](#) (int \_\_ix)
  - char [narrow](#) (char\_type \_\_c, char \_\_default) const
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \* \_\_p)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
  - [\\_\\_stringbuf\\_type](#) \* [rdbuf](#) () const
  - [iostate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [\\_\\_ostream\\_type](#) & [seekp](#) (pos\_type)
  - [\\_\\_ostream\\_type](#) & [seekp](#) (off\_type, [ios\\_base::seekdir](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [\\_\\_string\\_type](#) [str](#) () const
  - void [str](#) (const [\\_\\_string\\_type](#) & \_\_s)
  - pos\_type [tellp](#) ()
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - char\_type [widen](#) (char \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) & (\* \_\_pf) ([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) & (\* \_\_pf) ([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) & (\* \_\_pf) ([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)

- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [operator void \\*](#) () const
- bool [operator!](#) () const

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)

- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

#### Protected Types

- enum { `_S_local_word_size` }

#### Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename `_ValueT` >  
`_ostream_type` & `_M_insert` (`_ValueT` \_\_v)
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

#### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`,  
`_Traits` > \* `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

## 4.626.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_ostringstream< _CharT, _Traits, _Alloc >
```

Controlling output for std::string.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 387 of file `sstream`.

## 4.626.2 Member Typedef Documentation

4.626.2.1 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type` [inherited]

These are non-standard types.

Definition at line 90 of file `basic_ios.h`.

4.626.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i)` [inherited]

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.626.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`

- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file `ios_base.h`.

#### 4.626.2.4 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file `ios_base.h`.

#### 4.626.2.5 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file `ios_base.h`.

## 4.626.2.6 typedef \_Ios\_Seekdir std::ios\_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

## 4.626.3 Member Enumeration Documentation

## 4.626.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

## 4.626.4 Constructor &amp; Destructor Documentation

## 4.626.4.1 template&lt;typename \_CharT, typename \_Traits, typename \_Alloc&gt; std::basic\_ostringstream&lt;\_CharT, \_Traits, \_Alloc&gt;::basic\_ostringstream ( ios\_base::openmode \_\_mode = ios\_base::out ) [inline], [explicit]

Default constructor starts with an empty string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::out` is automatically included in `mode`.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 423 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

## 4.626.4.2 template&lt;typename \_CharT, typename \_Traits, typename \_Alloc&gt; std::basic\_ostringstream&lt;\_CharT, \_Traits, \_Alloc&gt;::basic\_ostringstream ( const \_\_string\_type &amp; \_\_str, ios\_base::openmode \_\_mode = ios\_base::out ) [inline], [explicit]

Starts with an existing string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::out` is automatically included in `mode`.

Initializes `sb` using `str` and `mode|out`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 441 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

**4.626.4.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream<_CharT, _Traits, _Alloc>::~~basic_ostringstream( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 452 of file `sstream`.

#### 4.626.5 Member Function Documentation

**4.626.5.1** `const locale& std::ios_base::M_getloc( ) const [inline],[inherited]`

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

**4.626.5.2** `template<typename _CharT, typename _Traits> void std::basic_ostream<_CharT, _Traits>::M_write( const char.type * __s, streamsize __n ) [inline],[inherited]`

Core write functionality, without sentry.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 311 of file `ostream`.

Referenced by `std::basic_ostream<_CharT, _Traits>::write()`.

**4.626.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad( ) const [inline],[inherited]`

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

**4.626.5.4** `template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.626.5.5** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of \_\_rhs into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

References std::basic\_ios< \_CharT, \_Traits >::exceptions(), std::basic\_ios< \_CharT, \_Traits >::fill(), std::ios\_base::flags(), std::ios\_base::getloc(), std::ios\_base::precision(), std::basic\_ios< \_CharT, \_Traits >::tie(), std::tie(), and std::ios\_base::width().

**4.626.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic\_ios.h.

4.626.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const`  
`[inline], [inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

4.626.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except )`  
`[inline], [inherited]`

Throwing exceptions on errors.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file `basic_ios.h`.

4.626.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const`  
`[inline], [inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Definition at line 195 of file `basic_ios.h`.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**4.626.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const`  
`[inline], [inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 364 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::basic\_ios< char, \_Traits >::fill().

**4.626.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )`  
`[inline], [inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file basic\_ios.h.

**4.626.5.12** `fmtflags std::ios_base::flags ( ) const` `[inline], [inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.626.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` `[inline], [inherited]`

Setting new format flags all at once.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

**4.626.5.14** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::flush ( ) [inherited]`

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns `-1`, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**4.626.5.15** `locale std::ios_base::getloc ( ) const [inline], [inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::ws()`.

**4.626.5.16** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good ( ) const [inline], [inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**4.626.5.17** `template<typename _CharT, typename _Traits> locale basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 114 of file `basic_ios.tcc`.

References `std::ios_base::imbue()`.

Referenced by `std::operator<<()`.

**4.626.5.18** `template<typename _CharT, typename _Traits> void basic_ios<_CharT, _Traits>::init ( basic_streambuf<_CharT, _Traits> * __sb ) [protected], [inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`, `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`, `std::basic_ios<_CharT, _Traits>::basic_ios()`, `std::basic_istream<_CharT, _Traits>::basic_istream()`, `std::basic_istream<_CharT, _Traits, _Alloc>::basic_istream()`, `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`, `std::basic_ostream<_CharT, _Traits>::basic_ostream()`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream()`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream()`.

**4.626.5.19** `long& std::ios_base::iword ( int __ix ) [inline], [inherited]`

Access to integer array.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

4.626.5.20 `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char __dfault ) const [inline], [inherited]`

Squeezes characters.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file `basic_ios.h`.

4.626.5.21 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

4.626.5.22 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file `basic_ios.h`.

4.626.5.23 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

4.626.5.24 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type &(*)(__ios_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

4.626.5.25 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `ioanip` header.

Definition at line 127 of file `ostream`.

4.626.5.26 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

4.626.5.27 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

4.626.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( bool __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.626.5.29 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( short __n ) [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

4.626.5.30 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( unsigned short __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.626.5.31 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( int __n ) [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

4.626.5.32 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( unsigned int __n ) [inline],[inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file `ostream`.

4.626.5.33 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( long long __n ) [inline],[inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

4.626.5.34 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( unsigned long long __n ) [inline],[inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

4.626.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits>::operator<<( double __f ) [inline],[inherited]`

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

**4.626.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( float __f ) [inline],[inherited]`

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**4.626.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long double __f ) [inline],[inherited]`

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**4.626.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline],[inherited]`

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.626.5.39** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

**4.626.5.40** `streamsize std::ios_base::precision( ) const [inline],[inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**4.626.5.41** `streamsize std::ios_base::precision( streamsize __prec ) [inline],[inherited]`

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().

Definition at line 630 of file ios\_base.h.

4.626.5.42 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::put ( char_type __c ) [inherited]`

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

4.626.5.43 `void*& std::ios_base::pword ( int __ix ) [inline], [inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

4.626.5.44 `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf ( basic_streambuf< _CharT, _Traits > * __sb ) [inherited]`

Changing the underlying buffer.

## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.626.5.45** `template<typename _CharT, typename _Traits, typename _Alloc> __stringbuf_type* std::basic_ostringstream<_CharT, _Traits, _Alloc>::rdbuf( ) const [inline]`

Accessing the underlying buffer.

## Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 463 of file `sstream`.

**4.626.5.46** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline], [inherited]`

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**4.626.5.47** `void std::ios_base::register_callback( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.626.5.48** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::seekp ( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

**4.626.5.49** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

**4.626.5.50** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline], [inherited]`

Setting new format flags.

#### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**4.626.5.51** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

#### Parameters

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

#### Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios\_base.h.

**4.626.5.52** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline]`, `[inherited]`

Sets additional flags in the error state.

#### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file basic\_ios.h.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::tr2::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unset()`, and `std::ws()`.

**4.626.5.53** `template<typename _CharT, typename _Traits, typename _Alloc> __string_type std::basic_ostringstream< _CharT, _Traits, _Alloc >::str ( ) const` `[inline]`

Copying out the string buffer.

**Returns**

`rdbuf() -> str()`

Definition at line 471 of file sstream.

Referenced by `std::operator<<()`.

4.626.5.54 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_ostringstream<_CharT, _Traits, _Alloc>::str( const __string_type & __s ) [inline]`

Setting a new buffer.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf() -> str(s)`.

Definition at line 481 of file sstream.

4.626.5.55 `static bool std::ios_base::sync_with_stdio( bool __sync = true ) [static], [inherited]`

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

4.626.5.56 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type basic_ostream<_CharT, _Traits>::tellp( ) [inherited]`

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

Definition at line 237 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

4.626.5.57 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( ) const [inline], [inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.626.5.58** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( basic_ostream< _CharT, _Traits > * __tiestr ) [inline],[inherited]`

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 301 of file `basic_ios.h`.

**4.626.5.59** `void std::ios_base::unsetf ( fmtflags __mask ) [inline],[inherited]`

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**4.626.5.60** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen ( char __c ) const [inline],[inherited]`

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (> (getloc())).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios<char, \_Traits>::fill(), std::basic\_istream<char>::get(), std::basic\_istream<char>::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

**4.626.5.61 streamsize std::ios\_base::width ( ) const** [inline], [inherited]

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::operator>>().

**4.626.5.62 streamsize std::ios\_base::width ( streamsize \_\_wide )** [inline], [inherited]

Changing flags.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

#### Returns

The previous value of width().

Definition at line 653 of file ios\_base.h.

**4.626.5.63 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits> & basic\_ostream<\_CharT, \_Traits>::write ( const char\_type \* \_\_s, streamsize \_\_n )** [inherited]

Character string insertion.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

#### Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted

- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic\_ostream< \_CharT, \_Traits >::\_M\_write(), and std::ios\_base::badbit.

**4.626.5.64** static int std::ios\_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

### 4.626.6 Member Data Documentation

**4.626.6.1** const fmtflags std::ios\_base::adjustfield [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of setf.

Definition at line 310 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::internal(), std::left(), and std::right().

**4.626.6.2** const openmode std::ios\_base::app [static], [inherited]

Seek to end before each write.

Definition at line 364 of file ios\_base.h.

**4.626.6.3** const openmode std::ios\_base::ate [static], [inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**4.626.6.4** const iostate std::ios\_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(),

std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

#### 4.626.6.5 const fmtflags std::ios\_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 313 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 4.626.6.6 const seekdir std::ios\_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

#### 4.626.6.7 const openmode std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 4.626.6.8 const fmtflags std::ios\_base::boolalpha [static], [inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

#### 4.626.6.9 const seekdir std::ios\_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 4.626.6.10 const fmtflags std::ios\_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

**4.626.6.11 const seekdir std::ios\_base::end** [static],[inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**4.626.6.12 const iostate std::ios\_base::eofbit** [static],[inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsomewhat(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unset(), and std::ws().

**4.626.6.13 const iostate std::ios\_base::failbit** [static],[inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.626.6.14 const fmtflags std::ios\_base::fixed** [static],[inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios\_base.h.

Referenced by std::fixed().

**4.626.6.15 const fmtflags std::ios\_base::floatfield** [static],[inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 316 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

**4.626.6.16 const iostate std::ios\_base::goodbit** [static],[inherited]

Indicates all is well.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsomewhat(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unset().

#### 4.626.6.17 const fmtflags std::ios\_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 4.626.6.18 const openmode std::ios\_base::in [static], [inherited]

Open for input. Default for ifstream and fstream.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

#### 4.626.6.19 const fmtflags std::ios\_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 272 of file ios\_base.h.

Referenced by std::internal().

#### 4.626.6.20 const fmtflags std::ios\_base::left [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

#### 4.626.6.21 const fmtflags std::ios\_base::oct [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**4.626.6.22 const openmode std::ios\_base::out** [static],[inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**4.626.6.23 const fmtflags std::ios\_base::right** [static],[inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.626.6.24 const fmtflags std::ios\_base::scientific** [static],[inherited]

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.626.6.25 const fmtflags std::ios\_base::showbase** [static],[inherited]

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.626.6.26 const fmtflags std::ios\_base::showpoint** [static],[inherited]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.626.6.27 const fmtflags std::ios\_base::showpos** [static],[inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.626.6.28 const fmtflags std::ios\_base::skipws** [static],[inherited]

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::skipws()`.

**4.626.6.29 const openmode std::ios\_base::trunc** [static],[inherited]

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.626.6.30** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::noinitbuf()`, `std::unitbuf()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**4.626.6.31** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outlter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [sstream](#)

## 4.627 `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- typedef `regex_constants::syntax_option_type` **flag\_type**
- typedef `traits_type::locale_type` **locale\_type**
- typedef `traits_type::string_type` **string\_type**
- typedef `_Rx_traits` **traits\_type**
- typedef `_Ch_type` **value\_type**

### Public Member Functions

- `basic_regex ()`
- `basic_regex (const _Ch_type * __p, flag_type __f=ECMAScript)`
- `basic_regex (const _Ch_type * __p, std::size_t __len, flag_type __f)`
- `basic_regex (const basic\_regex & __rhs)`
- `basic_regex (const basic\_regex && __rhs) noexcept`
- `template<typename _Ch_traits, typename _Ch_alloc >  
basic_regex (const std::basic\_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, flag_type __f=ECMAScript)`
- `template<typename _InputIterator >  
basic_regex (_InputIterator __first, _InputIterator __last, flag_type __f=ECMAScript)`
- `basic_regex (initializer_list< _Ch_type > __l, flag_type __f=ECMAScript)`
- `~basic_regex ()`
- `const __detail::_AutomatonPtr & M\_get\_automaton () const`
- `basic\_regex & assign (const basic\_regex & __rhs)`
- `basic\_regex & assign (basic\_regex && __rhs) noexcept`
- `basic\_regex & assign (const _Ch_type * __p, flag_type __flags=ECMAScript)`
- `basic\_regex & assign (const _Ch_type * __p, std::size_t __len, flag_type __flags)`
- `template<typename _Ch_traits, typename _Alloc >  
basic\_regex & assign (const basic\_string< _Ch_type, _Ch_traits, _Alloc > & __s, flag_type __flags=ECMAScript)`

- template<typename \_InputIterator >  
basic\_regex & assign (\_InputIterator \_\_first, \_InputIterator \_\_last, flag\_type \_\_flags=ECMAScript)
- basic\_regex & assign (initializer\_list<\_Ch\_type> \_\_l, flag\_type \_\_flags=ECMAScript)
- flag\_type flags () const
- locale\_type getloc () const
- locale\_type imbue (locale\_type \_\_loc)
- unsigned int mark\_count () const
- basic\_regex & operator= (const basic\_regex &\_\_rhs)
- basic\_regex & operator= (basic\_regex &&\_\_rhs) noexcept
- basic\_regex & operator= (const \_Ch\_type \*\_\_p)
- template<typename \_Ch\_traits, typename \_Alloc >  
basic\_regex & operator= (const basic\_string<\_Ch\_type, \_Ch\_traits, \_Alloc> &\_\_s)
- void swap (basic\_regex &\_\_rhs)

### Static Public Attributes

#### Constants

std [28.8.1](1)

- static constexpr flag\_type **icase**
- static constexpr flag\_type **nosubs**
- static constexpr flag\_type **optimize**
- static constexpr flag\_type **collate**
- static constexpr flag\_type **ECMAScript**
- static constexpr flag\_type **basic**
- static constexpr flag\_type **extended**
- static constexpr flag\_type **awk**
- static constexpr flag\_type **grep**
- static constexpr flag\_type **egrep**

### Protected Attributes

- \_\_detail::AutomatonPtr \_M\_automaton
- flag\_type \_M\_flags
- \_Rx\_traits \_M\_traits

### 4.627.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::basic_regex<_Ch_type, _Rx_traits>
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 335 of file regex.h.

### 4.627.2 Constructor & Destructor Documentation

4.627.2.1 template<typename \_Ch\_type, typename \_Rx\_traits = regex\_traits<\_Ch\_type>> std::basic\_regex<\_Ch\_type, \_Rx\_traits>::basic\_regex ( ) [inline]

Constructs a basic regular expression that does not match any character sequence.

Definition at line 367 of file regex.h.

4.627.2.2 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits>::basic_regex( const _Ch_type* __p, flag_type __f = ECMAScript ) [inline], [explicit]`

Constructs a basic regular expression from the sequence [`__p`, `__p + char_traits<_Ch_type>::length(__p)`) interpreted according to the flags in `__f`.

#### Parameters

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to the start of a C-style null-terminated string containing a regular expression. |
| <code>__f</code> | Flags indicating the syntax rules and options.                                              |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 385 of file `regex.h`.

4.627.2.3 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits>::basic_regex( const _Ch_type* __p, std::size_t __len, flag_type __f ) [inline]`

Constructs a basic regular expression from the sequence [`p`, `p + len`) interpreted according to the flags in `f`.

#### Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <code>__p</code>   | A pointer to the start of a string containing a regular expression. |
| <code>__len</code> | The length of the string containing the regular expression.         |
| <code>__f</code>   | Flags indicating the syntax rules and options.                      |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 403 of file `regex.h`.

4.627.2.4 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits>::basic_regex( const basic_regex<_Ch_type, _Rx_traits> &__rhs ) [inline]`

Copy-constructs a basic regular expression.

#### Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A <code>regex</code> object. |
|--------------------|------------------------------|

Definition at line 413 of file `regex.h`.

4.627.2.5 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits>::basic_regex( const basic_regex<_Ch_type, _Rx_traits> &&__rhs ) [inline], [noexcept]`

Move-constructs a basic regular expression.

#### Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A <code>regex</code> object. |
|--------------------|------------------------------|

Definition at line 423 of file `regex.h`.

4.627.2.6 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_traits ,  
typename _Ch_alloc > std::basic_regex<_Ch_type, _Rx_traits>::basic_regex ( const std::basic_string<  
_Ch_type, _Ch_traits, _Ch_alloc > &__s, flag_type __f=ECMAScript ) [inline], [explicit]`

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__s</code> | A string containing a regular expression.      |
| <code>__f</code> | Flags indicating the syntax rules and options. |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 439 of file `regex.h`.

4.627.2.7 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _InputIterator >  
std::basic_regex<_Ch_type, _Rx_traits>::basic_regex ( _InputIterator __first, _InputIterator __last, flag_type __f =  
ECMAScript ) [inline]`

Constructs a basic regular expression from the range `[first, last)` interpreted according to the flags in `f`.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__f</code>     | The format flags of the regular expression.                 |

#### Exceptions

|                          |                                                                      |
|--------------------------|----------------------------------------------------------------------|
| <code>regex_error</code> | if <code>[__first, __last)</code> is not a valid regular expression. |
|--------------------------|----------------------------------------------------------------------|

Definition at line 461 of file `regex.h`.

4.627.2.8 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,  
_Rx_traits>::basic_regex ( initializer_list<_Ch_type> __l, flag_type __f=ECMAScript ) [inline]`

Constructs a basic regular expression from an initializer list.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__l</code> | The initializer list.                       |
| <code>__f</code> | The format flags of the regular expression. |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 475 of file `regex.h`.

4.627.2.9 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits >::~~basic_regex ( ) [inline]`

Destroys a basic regular expression.

Definition at line 485 of file regex.h.

#### 4.627.3 Member Function Documentation

4.627.3.1 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign ( const basic_regex< _Ch_type, _Rx_traits > & __rhs ) [inline]`

the real assignment operator.

##### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

Definition at line 531 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::operator=()`.

4.627.3.2 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign ( basic_regex< _Ch_type, _Rx_traits > && __rhs ) [inline], [noexcept]`

The move-assignment operator.

##### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

Definition at line 544 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

4.627.3.3 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign ( const _Ch_type * __p, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

##### Parameters

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style null-terminated string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                                                                   |

##### Exceptions

|                          |                                                                                                                                                                                                         |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> .<br>If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 565 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

4.627.3.4 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign ( const _Ch_type * __p, std::size_t __len, flag_type __flags ) [inline]`

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

#### Parameters

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style string containing a regular expression pattern. |
| <code>__len</code>   | The length of the regular expression pattern string.                   |
| <code>__flags</code> | Syntax option flags.                                                   |

#### Exceptions

|                          |                                                                                                                                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> .<br>If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 582 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

4.627.3.5 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_type_traits , typename _Alloc > basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign ( const basic_string<_Ch_type, _Ch_type_traits, _Alloc > & __s, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__s</code>     | A string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                              |

#### Exceptions

|                          |                                                                                                                                                                                                         |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> .<br>If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 598 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

4.627.3.6 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _InputIterator > basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign ( _InputIterator __first, _InputIterator __last, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__flags</code> | Syntax option flags.                                        |

## Exceptions

|                    |                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex_error</i> | if p does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, the object remains unchanged. |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 621 of file regex.h.

References std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

Referenced by std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

4.627.3.7 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign ( initializer_list<_Ch_type> __l, flag_type __flags = ECMAScript ) [inline]`

Assigns a new regular expression to a regex object.

## Parameters

|                |                                                        |
|----------------|--------------------------------------------------------|
| <i>__l</i>     | An initializer list representing a regular expression. |
| <i>__flags</i> | Syntax option flags.                                   |

## Exceptions

|                    |                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex_error</i> | if __l does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, the object remains unchanged. |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 637 of file regex.h.

References std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

Referenced by std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

4.627.3.8 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> flag_type std::basic_regex<_Ch_type, _Rx_traits>::flags ( ) const [inline]`

Gets the flags used to construct the regular expression or in the last call to assign().

Definition at line 654 of file regex.h.

Referenced by std::basic\_regex< \_Ch\_type, \_Rx\_traits >::operator=().

4.627.3.9 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex<_Ch_type, _Rx_traits>::getloc ( ) const [inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 672 of file regex.h.

4.627.3.10 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex<_Ch_type, _Rx_traits>::imbue ( locale_type __loc ) [inline]`

Imbues the regular expression object with the given locale.

## Parameters

|              |           |
|--------------|-----------|
| <i>__loc</i> | A locale. |
|--------------|-----------|

Definition at line 664 of file regex.h.

4.627.3.11 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> unsigned int std::basic_regex<_Ch_type, _Rx_traits>::mark_count ( ) const [inline]`

Gets the number of marked subexpressions within the regular expression.

Definition at line 646 of file regex.h.

4.627.3.12 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator=( const basic_regex<_Ch_type, _Rx_traits> & _rhs ) [inline]`

Assigns one regular expression to another.

Definition at line 492 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

4.627.3.13 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator=( basic_regex<_Ch_type, _Rx_traits> && _rhs ) [inline], [noexcept]`

Move-assigns one regular expression to another.

Definition at line 499 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

4.627.3.14 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator=( const _Ch_type * __p ) [inline]`

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

#### Parameters

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to the start of a null-terminated C-style string containing a regular expression. |
|------------------|---------------------------------------------------------------------------------------------|

Definition at line 510 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

4.627.3.15 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_type_traits, typename _Alloc > basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::operator=( const basic_string<_Ch_type, _Ch_type_traits, _Alloc> & __s ) [inline]`

Replaces a regular expression with a new one constructed from a string.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__s</code> | A pointer to a string containing a regular expression. |
|------------------|--------------------------------------------------------|

Definition at line 521 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

4.627.3.16 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> void std::basic_regex<_Ch_type, _Rx_traits>::swap ( basic_regex<_Ch_type, _Rx_traits> & _rhs ) [inline]`

Swaps the contents of two regular expression objects.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

Definition at line 682 of file regex.h.

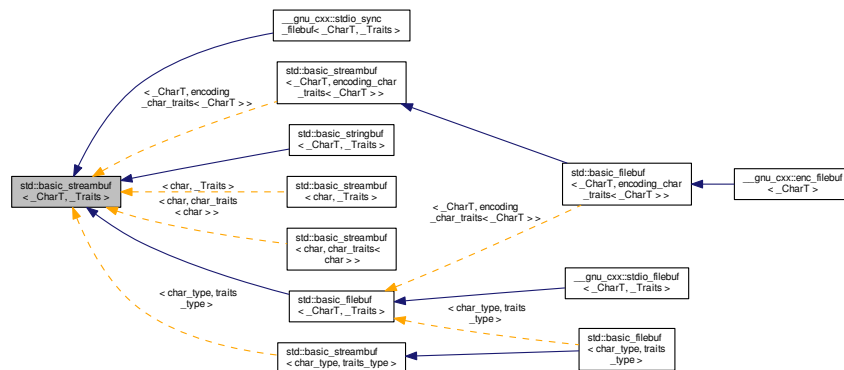
Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::swap()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.628 std::basic\_streambuf&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for `std::basic_streambuf< _CharT, _Traits >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_Traits` `traits_type`
- typedef `traits_type::int_type` `int_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef `traits_type::off_type` `off_type`
- typedef `basic_streambuf< char_type, traits_type >` `__streambuf_type`

## Public Member Functions

- virtual `~basic_streambuf()`
- `locale getloc()` const
- `streamsize in_avail()`
- `locale pubimbue(const locale &__loc)`
- `int_type sbumpc()`
- `int_type sgetc()`
- `streamsize sgetn(char_type *__s, streamsize __n)`

- [int\\_type snextc](#) ()
- [int\\_type sputbackc](#) ([char\\_type](#) \_\_c)
- [int\\_type sputc](#) ([char\\_type](#) \_\_c)
- [streamsize sputn](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [int\\_type sungetc](#) ()
- [basic\\_streambuf](#) \* [pubsetbuf](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [pos\\_type](#) [pubseekoff](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [pos\\_type](#) [pubseekpos](#) ([pos\\_type](#) \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [int](#) [pubsync](#) ()

#### Protected Member Functions

- [basic\\_streambuf](#) ()
- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
- void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) &\_\_loc)
- virtual [int\\_type](#) [overflow](#) ([int\\_type](#) \_\_c=[traits\\_type::eof](#)())
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=[traits\\_type::eof](#)())
- void [pbump](#) (int \_\_n)
- virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#), [ios\\_base::seekdir](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [basic\\_streambuf](#)
  - < [char\\_type](#), [\\_Traits](#) > \* [setbuf](#) ([char\\_type](#) \*, [streamsize](#))
- void [setg](#) ([char\\_type](#) \*\_\_gbeg, [char\\_type](#) \*\_\_gnext, [char\\_type](#) \*\_\_gend)
- void [setp](#) ([char\\_type](#) \*\_\_pbeg, [char\\_type](#) \*\_\_pend)
- virtual [streamsize](#) [showmanyc](#) ()
- virtual [int](#) [sync](#) ()
- virtual [int\\_type](#) [uflow](#) ()
- virtual [int\\_type](#) [underflow](#) ()
- virtual [streamsize](#) [xsgetn](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- virtual [streamsize](#) [xspn](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [char\\_type](#) \* [eback](#) () const
- [char\\_type](#) \* [gptr](#) () const
- [char\\_type](#) \* [egptr](#) () const
- [char\\_type](#) \* [pbase](#) () const
- [char\\_type](#) \* [pptr](#) () const
- [char\\_type](#) \* [epptr](#) () const

#### Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)

## Friends

- template<bool \_IsMove, typename \_CharT2 >  
     \_\_gnu\_cxx::\_\_enable\_if  
     < \_\_is\_char< \_CharT2 >  
     ::\_\_value, \_CharT2 \* >::\_\_type \_\_copy\_move\_a2 (istreambuf\_iterator< \_CharT2 >, istreambuf\_iterator< \_CharT2 >, \_CharT2 \*)
- streamsize \_\_copy\_streambufs\_eof (basic\_streambuf \*, basic\_streambuf \*, bool &)
- class basic\_ios< char\_type, traits\_type >
- class basic\_istream< char\_type, traits\_type >
- class basic\_ostream< char\_type, traits\_type >
- template<typename \_CharT2 >  
     \_\_gnu\_cxx::\_\_enable\_if  
     < \_\_is\_char< \_CharT2 >  
     ::\_\_value, istreambuf\_iterator  
     < \_CharT2 > >::\_\_type find (istreambuf\_iterator< \_CharT2 >, istreambuf\_iterator< \_CharT2 >, const \_CharT2 &)
- template<typename \_CharT2, typename \_Traits2, typename \_Alloc >  
     basic\_istream< \_CharT2,  
     \_Traits2 > & getline (basic\_istream< \_CharT2, \_Traits2 > &, basic\_string< \_CharT2, \_Traits2, \_Alloc > &, \_CharT2)
- class istreambuf\_iterator< char\_type, traits\_type >
- template<typename \_CharT2, typename \_Traits2 >  
     basic\_istream< \_CharT2,  
     \_Traits2 > & operator>> (basic\_istream< \_CharT2, \_Traits2 > &, \_CharT2 \*)
- template<typename \_CharT2, typename \_Traits2, typename \_Alloc >  
     basic\_istream< \_CharT2,  
     \_Traits2 > & operator>> (basic\_istream< \_CharT2, \_Traits2 > &, basic\_string< \_CharT2, \_Traits2, \_Alloc > &)
- class ostreambuf\_iterator< char\_type, traits\_type >

## 4.628.1 Detailed Description

```
template<typename _CharT, typename _Traits>class std::basic_streambuf< _CharT, _Traits >
```

The actual work of input and output (interface).

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:

- The controlled input sequence can be not readable.
- The controlled output sequence can be not writable.
- The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.

- The controlled sequences can support operations *directly* to or from associated sequences.
  - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
- the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
- If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and *xnext* < *xend* for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and *xbeg* < *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext[-1]* shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and *xnext* < *xend* for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 120 of file `streambuf`.

#### 4.628.2 Member Typedef Documentation

4.628.2.1 `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type>  
std::basic_streambuf<_CharT, _Traits>::__streambuf_type`

This is a non-standard type.

Definition at line 138 of file `streambuf`.

4.628.2.2 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_streambuf<_CharT, _Traits  
>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 129 of file `streambuf`.

4.628.2.3 `template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_streambuf<_CharT, _Traits  
>::__int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 131 of file `streambuf`.

4.628.2.4 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_streambuf<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 133 of file `streambuf`.

4.628.2.5 `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_streambuf<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 132 of file `streambuf`.

4.628.2.6 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_streambuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 130 of file `streambuf`.

### 4.628.3 Constructor & Destructor Documentation

4.628.3.1 `template<typename _CharT, typename _Traits> virtual std::basic_streambuf<_CharT, _Traits>::~~basic_streambuf( ) [inline], [virtual]`

Destructor deallocates no buffer space.

Definition at line 197 of file `streambuf`.

4.628.3.2 `template<typename _CharT, typename _Traits> std::basic_streambuf<_CharT, _Traits>::basic_streambuf( ) [inline], [protected]`

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the `basic_streambuf` class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 463 of file `streambuf`.

### 4.628.4 Member Function Documentation

4.628.4.1 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback( ) const [inline], [protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence

- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback(), std::basic\_streambuf< char\_type, traits\_type >::sputbackc(), and std::basic\_streambuf< char\_type, traits\_type >::sungetc().

**4.628.4.2** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::egptr ( )  
const [inline], [protected]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), std::basic\_streambuf< char\_type, traits\_type >::in\_avail(), std::basic\_streambuf< char\_type, traits\_type >::sbumpc(), std::basic\_streambuf< char\_type, traits\_type >::sgetc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str().

**4.628.4.3** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::eptr ( )  
const [inline], [protected]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::sputc().

**4.628.4.4** template<typename \_CharT, typename \_Traits> void std::basic\_streambuf< \_CharT, \_Traits >::gbump ( int \_\_n )  
[inline], [protected]

Moving the read position.

Parameters

|     |                             |
|-----|-----------------------------|
| __n | The delta by which to move. |
|-----|-----------------------------|

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::sbumpc(), std::basic\_streambuf< char\_type, traits\_type >::sputbackc(), std::basic\_streambuf< char\_type, traits\_type >::sungetc(), and std::basic\_streambuf< char\_type, traits\_type >::uflow().

**4.628.4.5** template<typename \_CharT, typename \_Traits> locale std::basic\_streambuf< \_CharT, \_Traits >::getloc ( ) const [inline]

Locale access.

#### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::pubimbue().

**4.628.4.6** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::gptr ( ) const [inline], [protected]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback(), std::basic\_streambuf< char\_type, traits\_type >::in\_avail(), std::basic\_streambuf< char\_type, traits\_type >::sbumpc(), std::basic\_streambuf< char\_type, traits\_type >::sgetc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_streambuf< char\_type, traits\_type >::sputbackc(), std::basic\_streambuf< char\_type, traits\_type >::sungetc(), and std::basic\_streambuf< char\_type, traits\_type >::uflow().

**4.628.4.7** template<typename \_CharT, typename \_Traits> virtual void std::basic\_streambuf< \_CharT, \_Traits >::imbue ( const locale & \_\_loc ) [inline], [protected], [virtual]

Changes translations.

#### Parameters

|       |               |
|-------|---------------|
| __loc | A new locale. |
|-------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 576 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

**4.628.4.8** `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( )`  
`[inline]`

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

**4.628.4.9** `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow ( int_type __c = traits_type::eof() )` `[inline], [protected], [virtual]`

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#), and [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 768 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#).

4.628.4.10 `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf<_CharT, _Traits>::pbackfail( int_type __c = traits_type::eof() ) [inline], [protected], [virtual]`

Tries to back up the input sequence.

#### Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

#### Returns

`eof()` on failure, *some other value* on success

#### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, and `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 724 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputbackc()`, and `std::basic_streambuf<char_type, traits_type>::sungetc()`.

4.628.4.11 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::pbase( ) const [inline], [protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 529 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

4.628.4.12 `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::pbump( int __n ) [inline], [protected]`

Moving the write position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::sputc().

4.628.4.13 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( )  
const [inline], [protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::sputc(), and std::basic\_stringbuf< \_CharT, \_Traits, \_-Alloc >::str().

4.628.4.14 `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::pubimbue ( const  
locale & __loc ) [inline]`

Entry point for imbue().

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 209 of file streambuf.

4.628.4.15 `template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (   
off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline]`

Alters the stream position.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__off</code>  | Offset.                       |
| <code>__way</code>  | Value for ios_base::seekdir.  |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.628.4.16 `template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos ( pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline]`

Alters the stream position.

#### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

Definition at line 263 of file `streambuf`.

4.628.4.17 `template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf ( char_type * __s, streamsize __n ) [inline]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file `streambuf`.

4.628.4.18 `template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline]`

Calls virtual `sync` function.

Definition at line 271 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

4.628.4.19 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline]`

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::istreambuf_iterator< _CharT, _Traits >::operator++()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

4.628.4.20 `template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff ( off_type , ios_base::seekdir , ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 602 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubseekoff()`.

```
4.628.4.21 template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf< _CharT, _Traits
>::seekpos(pos_type , ios_base::openmode = ios_base::in | ios_base::out) [inline],
[protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 614 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubseekpos()`.

```
4.628.4.22 template<typename _CharT, typename _Traits> virtual basic_streambuf<char_type, Traits>*
std::basic_streambuf< _CharT, _Traits >::setbuf(char_type *, streamsize) [inline],
[protected], [virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 591 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubsetbuf()`.

```
4.628.4.23 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setg(char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline], [protected]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 509 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**4.628.4.24** `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setp ( char_type * __pbeg, char_type * __pend )` `[inline]`, `[protected]`

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 555 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**4.628.4.25** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( )` `[inline]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**4.628.4.26** `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn ( char_type * __s, streamsize __n )` `[inline]`

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.628.4.27 `template<typename _CharT, typename _Traits> virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ( ) [inline], [protected], [virtual]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 649 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::in_avail()`.

4.628.4.28 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline]`

Getting the next character.

#### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.628.4.29 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputbackc ( char_type __c ) [inline]`

Pushing characters back into the input stream.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 372 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**4.628.4.30** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c ) [inline]`

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(-__c)`.

Definition at line 424 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**4.628.4.31** `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * __s, streamsize __n ) [inline]`

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 450 of file `streambuf`.

**4.628.4.32** `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline]`

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 397 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::unget()`.

**4.628.4.33** `template<typename _CharT, typename _Traits> virtual int std::basic_streambuf< _CharT, _Traits >::sync ( void )`  
`[inline], [protected], [virtual]`

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 627 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubsync()`.

**4.628.4.34** `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow (`  
`) [inline], [protected], [virtual]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 700 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sbumpc()`.

**4.628.4.35** `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits`  
`>::underflow ( ) [inline], [protected], [virtual]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, and `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 687 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sgetc()`, and `std::basic_streambuf< char_type, traits_type >::uflow()`.

**4.628.4.36** `template<typename _CharT, typename _Traits> streamsize basic_streambuf< _CharT, _Traits >::xsgetn (char_type * __s, streamsize __n) [protected], [virtual]`

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 46 of file `streambuf.tcc`.

References `std::min()`.

Referenced by `std::basic_streambuf< char_type, traits_type >::xsgetn()`.

**4.628.4.37** `template<typename _CharT, typename _Traits> streamsize basic_streambuf< _CharT, _Traits >::xspun ( const char_type * __s, streamsize __n) [protected], [virtual]`

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 80 of file `streambuf.tcc`.

References `std::min()`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputn()`.

## 4.628.5 Member Data Documentation

**4.628.5.1** `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::M_buf_locale`  
[protected]

Current locale setting.

Definition at line 192 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`, `std::basic_streambuf< char_type, traits_type >::getloc()`, and `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**4.628.5.2** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_in_beg`  
[protected]

Start of get area.

Definition at line 184 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::eback()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**4.628.5.3** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_in_cur`  
[protected]

Current read area.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::gbump()`, `std::basic_streambuf< char_type, traits_type >::gptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**4.628.5.4** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_in_end`  
[protected]

End of get area.

Definition at line 186 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**4.628.5.5** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_out_beg`  
[protected]

Start of put area.

Definition at line 187 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbase()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**4.628.5.6** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_out_cur`  
[protected]

Current put area.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbump()`, `std::basic_streambuf< char_type, traits_type >::pptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**4.628.5.7** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::M_out_end`  
[protected]

End of put area.

Definition at line 189 of file `streambuf`.

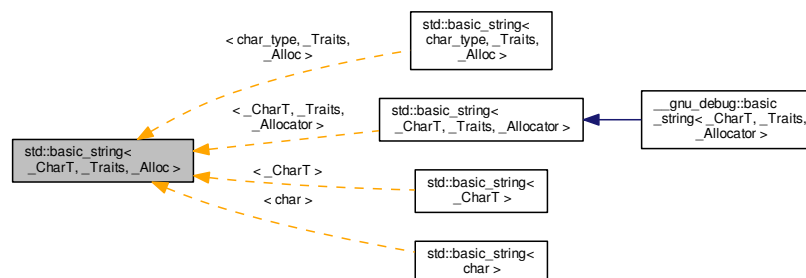
Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

The documentation for this class was generated from the following files:

- [streambuf](#)
- [streambuf.tcc](#)

## 4.629 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference

Inheritance diagram for `std::basic_string< _CharT, _Traits, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic\_string>` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator>` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic\_string>` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator>` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- [basic\\_string](#) ()
- [basic\\_string](#) (const `_Alloc` &\_\_a)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const `_Alloc` &\_\_a)
- [basic\\_string](#) (const `_CharT` \*\_\_s, size\_type \_\_n, const `_Alloc` &\_\_a=`_Alloc`())
- [basic\\_string](#) (const `_CharT` \*\_\_s, const `_Alloc` &\_\_a=`_Alloc`())
- [basic\\_string](#) (size\_type \_\_n, `_CharT` \_\_c, const `_Alloc` &\_\_a=`_Alloc`())
- [basic\\_string](#) ([basic\\_string](#) &&\_\_str) noexcept
- [basic\\_string](#) ([initializer\\_list](#)< `_CharT` > \_\_l, const `_Alloc` &\_\_a=`_Alloc`())
- template<class `_InputIterator` >  
  [basic\\_string](#) (`_InputIterator` \_\_beg, `_InputIterator` \_\_end, const `_Alloc` &\_\_a=`_Alloc`())
- [~basic\\_string](#) () noexcept
- template<typename `_InIterator` >  
  `_CharT` \* **S\_construct** (`_InIterator` \_\_beg, `_InIterator` \_\_end, const `_Alloc` &\_\_a, [forward\\_iterator\\_tag](#))
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & **append** (const `_CharT` \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **append** (const `_CharT` \*\_\_s)
- [basic\\_string](#) & **append** (size\_type \_\_n, `_CharT` \_\_c)
- [basic\\_string](#) & **append** ([initializer\\_list](#)< `_CharT` > \_\_l)
- template<class `_InputIterator` >  
  [basic\\_string](#) & **append** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)

- `basic_string` & `assign` (const `basic_string` &\_\_str)
- `basic_string` & `assign` (`basic_string` &&\_\_str)
- `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- `basic_string` & `assign` (const `_CharT` \* \_\_s, size\_type \_\_n)
- `basic_string` & `assign` (const `_CharT` \* \_\_s)
- `basic_string` & `assign` (size\_type \_\_n, `_CharT` \_\_c)
- `template<class _InputIterator>`  
`basic_string` & `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `basic_string` & `assign` (`initializer_list`< `_CharT` > \_\_l)
- `const_reference` `at` (size\_type \_\_n) const
- `reference` `at` (size\_type \_\_n)
- `reference` `back` ()
- `const_reference` `back` () const
- `iterator` `begin` () noexcept
- `const_iterator` `begin` () const noexcept
- `const _CharT` \* `c_str` () const noexcept
- size\_type `capacity` () const noexcept
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `cend` () const noexcept
- `void` `clear` () noexcept
- `int` `compare` (const `basic_string` &\_\_str) const
- `int` `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str) const
- `int` `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- `int` `compare` (const `_CharT` \* \_\_s) const
- `int` `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \* \_\_s) const
- `int` `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \* \_\_s, size\_type \_\_n2) const
- size\_type `copy` (`_CharT` \* \_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- `const_reverse_iterator` `crbegin` () const noexcept
- `const_reverse_iterator` `crend` () const noexcept
- `const _CharT` \* `data` () const noexcept
- `bool` `empty` () const noexcept
- `iterator` `end` () noexcept
- `const_iterator` `end` () const noexcept
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=`npos`)
- `iterator` `erase` (iterator \_\_position)
- `iterator` `erase` (iterator \_\_first, iterator \_\_last)
- size\_type `find` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find` (const `_CharT` \* \_\_s, size\_type \_\_pos=0) const
- size\_type `find` (`_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (`_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_of` (const `_CharT` \* \_\_s, size\_type \_\_pos=0) const
- size\_type `find_first_of` (`_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_last_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=`npos`) const noexcept
- size\_type `find_last_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const

- `size_type find_last_not_of` (const `_CharT *__s`, `size_type __pos=npos`) const
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=npos`) const noexcept
- `size_type find_last_of` (const `basic_string &__str`, `size_type __pos=npos`) const noexcept
- `size_type find_last_of` (const `_CharT *__s`, `size_type __pos`, `size_type __n`) const
- `size_type find_last_of` (const `_CharT *__s`, `size_type __pos=npos`) const
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npos`) const noexcept
- reference `front` ()
- const\_reference `front` () const
- allocator\_type `get_allocator` () const noexcept
- void `insert` (iterator `__p`, `size_type __n`, `_CharT __c`)
- template<class `_InputIterator` >  
void `insert` (iterator `__p`, `_InputIterator __beg`, `_InputIterator __end`)
- void `insert` (iterator `__p`, `initializer_list<_CharT> __l`)
- `basic_string & insert` (`size_type __pos1`, const `basic_string &__str`)
- `basic_string & insert` (`size_type __pos1`, const `basic_string &__str`, `size_type __pos2`, `size_type __n`)
- `basic_string & insert` (`size_type __pos`, const `_CharT *__s`, `size_type __n`)
- `basic_string & insert` (`size_type __pos`, const `_CharT *__s`)
- `basic_string & insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- iterator `insert` (iterator `__p`, `_CharT __c`)
- `size_type length` () const noexcept
- `size_type max_size` () const noexcept
- `basic_string & operator+=` (const `basic_string &__str`)
- `basic_string & operator+=` (const `_CharT *__s`)
- `basic_string & operator+=` (`_CharT __c`)
- `basic_string & operator+=` (`initializer_list<_CharT> __l`)
- `basic_string & operator=` (const `basic_string &__str`)
- `basic_string & operator=` (const `_CharT *__s`)
- `basic_string & operator=` (`_CharT __c`)
- `basic_string & operator=` (`basic_string &&__str`)
- `basic_string & operator=` (`initializer_list<_CharT> __l`)
- const\_reference `operator[]` (`size_type __pos`) const
- reference `operator[]` (`size_type __pos`)
- void `pop_back` ()
- void `push_back` (`_CharT __c`)
- `reverse_iterator rbegin` () noexcept
- const\_reverse\_iterator `rbegin` () const noexcept
- `reverse_iterator rend` () noexcept
- const\_reverse\_iterator `rend` () const noexcept
- `basic_string & replace` (`size_type __pos`, `size_type __n`, const `basic_string &__str`)
- `basic_string & replace` (`size_type __pos1`, `size_type __n1`, const `basic_string &__str`, `size_type __pos2`, `size_type __n2`)
- `basic_string & replace` (`size_type __pos`, `size_type __n1`, const `_CharT *__s`, `size_type __n2`)
- `basic_string & replace` (`size_type __pos`, `size_type __n1`, const `_CharT *__s`)
- `basic_string & replace` (`size_type __pos`, `size_type __n1`, `size_type __n2`, `_CharT __c`)
- `basic_string & replace` (iterator `__i1`, iterator `__i2`, const `basic_string &__str`)
- `basic_string & replace` (iterator `__i1`, iterator `__i2`, const `_CharT *__s`, `size_type __n`)
- `basic_string & replace` (iterator `__i1`, iterator `__i2`, const `_CharT *__s`)
- `basic_string & replace` (iterator `__i1`, iterator `__i2`, `size_type __n`, `_CharT __c`)
- template<class `_InputIterator` >  
`basic_string & replace` (iterator `__i1`, iterator `__i2`, `_InputIterator __k1`, `_InputIterator __k2`)
- `basic_string & replace` (iterator `__i1`, iterator `__i2`, `_CharT *__k1`, `_CharT *__k2`)

- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, [initializer\\_list](#)<\_CharT> \_\_l)
- void [reserve](#) (size\_type \_\_res\_arg=0)
- void [resize](#) (size\_type \_\_n, \_CharT \_\_c)
- void [resize](#) (size\_type \_\_n)
- size\_type [rfind](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const
- size\_type [rfind](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- [basic\\_string](#) [substr](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- void [swap](#) ([basic\\_string](#) &\_\_s)

#### Static Public Attributes

- static const size\_type [npos](#)

#### 4.629.1 Detailed Description

template<typename \_CharT, typename \_Traits, typename \_Alloc>class std::basic\_string<\_CharT, \_Traits, \_Alloc>

Managing sequences of characters and character-like objects.

#### Template Parameters

|                         |                                                                              |
|-------------------------|------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character                                                            |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> <_CharT>. |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <a href="#">allocator</a> <_CharT>.              |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only [push\\_back](#), [at](#), and array access are supported.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

[basic_string<char_type>]
_M_dataplus
_M_p ----->
 [_Rep]
 _M_length
 _M_capacity
 _M_refcount
 unnamed array of char_type
```

Where the [\\_M\\_p](#) points to the first character in the string, and you cast it to a pointer-to-[\\_Rep](#) and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: [\\_Rep::\\_M\\_data\(\)](#), and [string::\\_M\\_rep\(\)](#); and the allocation function which gets a block of raw bytes and with room enough and constructs a [\\_Rep](#) object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 112 of file `basic_string.h`.

#### 4.629.2 Constructor & Destructor Documentation

**4.629.2.1** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( ) [inline]`

Default constructor creates an empty string.

Definition at line 437 of file `basic_string.h`.

Referenced by `std::basic_string< char >::substr()`.

**4.629.2.2** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const _Alloc & _a ) [explicit]`

Construct an empty string using allocator *a*.

Definition at line 178 of file `basic_string.tcc`.

**4.629.2.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const basic_string< _CharT, _Traits, _Alloc > & _str )`

Construct string with copy of value of *str*.

##### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

Definition at line 170 of file `basic_string.tcc`.

**4.629.2.4** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string( const basic_string< _CharT, _Traits, _Alloc > & _str, size_type __pos, size_type __n = npos )`

Construct string as copy of a substring.

##### Parameters

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>__str</code> | Source string.                                    |
| <code>__pos</code> | Index of first character to copy from.            |
| <code>__n</code>   | Number of characters to copy (default remainder). |

Definition at line 184 of file `basic_string.tcc`.

4.629.2.5 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n, const _Alloc & __a )`

Construct string as copy of a substring.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

Definition at line 194 of file basic\_string.tcc.

4.629.2.6 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( const _CharT * __s, size_type __n, const _Alloc & __a = _Alloc() )`

Construct string initialized by a character array.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 206 of file basic\_string.tcc.

4.629.2.7 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( const _CharT * __s, const _Alloc & __a = _Alloc() )`

Construct string as copy of a C string.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source C string.                                 |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 213 of file basic\_string.tcc.

4.629.2.8 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( size_type __n, _CharT __c, const _Alloc & __a = _Alloc() )`

Construct string as multiple characters.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__n</code> | Number of characters.                            |
| <code>__c</code> | Character to use.                                |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 220 of file basic\_string.tcc.

4.629.2.9 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( basic_string< _CharT, _Traits, _Alloc > && __str ) [inline], [noexcept]`

Move construct string.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 507 of file `basic_string.h`.

4.629.2.10 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( initializer_list< _CharT > __l, const _Alloc & __a = _Alloc() )`

Construct string from an initializer list.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | std::initializer_list of characters.             |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 235 of file `basic_string.tcc`.

4.629.2.11 `template<typename _CharT, typename _Traits, typename _Alloc> template<typename _InputIterator > std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( _InputIterator __beg, _InputIterator __end, const _Alloc & __a = _Alloc() )`

Construct string as copy of a range.

#### Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

Definition at line 228 of file `basic_string.tcc`.

4.629.2.12 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ( ) [inline], [noexcept]`

Destroy the string instance.

Definition at line 538 of file `basic_string.h`.

### 4.629.3 Member Function Documentation

4.629.3.1 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append ( const basic_string< _CharT, _Traits, _Alloc > & __str )`

Append a string to this string.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 325 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::size(), and std::size().

Referenced by std::basic\_string< char >::append(), std::collate< \_CharT >::do\_transform(), std::operator+(), std::basic\_string< char >::operator+=( ), and std::operator>>().

**4.629.3.2** template<typename \_CharT, typename \_Traits, typename \_Alloc > **basic\_string**< \_CharT, \_Traits, \_Alloc > & **std::basic\_string**< \_CharT, \_Traits, \_Alloc >::append ( const **basic\_string**< \_CharT, \_Traits, \_Alloc > & \_\_str, size\_type \_\_pos, size\_type \_\_n )

Append a substring.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | The string to append.                          |
| <code>__pos</code> | Index of the first character of str to append. |
| <code>__n</code>   | The number of characters to append.            |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 342 of file basic\_string.tcc.

References std::size().

**4.629.3.3** template<typename \_CharT, typename \_Traits, typename \_Alloc > **basic\_string**< \_CharT, \_Traits, \_Alloc > & **std::basic\_string**< \_CharT, \_Traits, \_Alloc >::append ( const \_CharT \* \_\_s, size\_type \_\_n )

Append a C substring.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

**Returns**

Reference to this string.

Definition at line 298 of file basic\_string.tcc.

References std::size().

**4.629.3.4** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( const _CharT * __s ) [inline]`

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

Definition at line 1006 of file basic\_string.h.

**4.629.3.5** `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append ( size_type __n, _CharT __c )`

Append multiple characters.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__n</code> | The number of characters to append. |
| <code>__c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.

Definition at line 281 of file basic\_string.tcc.

References std::size().

**4.629.3.6** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( initializer_list< _CharT > __l ) [inline]`

Append an initializer\_list of characters.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__l</code> | The initializer_list of characters to append. |
|------------------|-----------------------------------------------|

**Returns**

Reference to this string.

Definition at line 1030 of file basic\_string.h.

Referenced by std::basic\_string< char >::append().

4.629.3.7 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append ( _InputIterator __first, _InputIterator __last ) [inline]`

Append a range of characters.

#### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

#### Returns

Reference to this string.

Appends characters in the range [`__first`,`__last`) to this string.

Definition at line 1044 of file basic\_string.h.

4.629.3.8 `template<typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign ( const basic_string< _CharT, _Traits, _Alloc > & __str )`

Set value to contents of another string.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

#### Returns

Reference to this string.

Definition at line 243 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::get\_allocator().

Referenced by std::basic\_string< char >::assign(), std::basic\_string< char >::operator=(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str().

4.629.3.9 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( basic_string< _CharT, _Traits, _Alloc > && __str ) [inline]`

Set value to contents of another string.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

#### Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 1079 of file basic\_string.h.

```
4.629.3.10 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
 _Traits, _Alloc >::assign (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n)
 [inline]
```

Set value to a substring of a string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 1100 of file `basic_string.h`.

Referenced by `std::basic_string< char >::assign()`.

```
4.629.3.11 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > &
 std::basic_string< _CharT, _Traits, _Alloc >::assign (const _CharT * __s, size_type __n)
```

Set value to a C substring.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

#### Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 259 of file `basic_string.tcc`.

References `std::size()`.

```
4.629.3.12 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
 _Traits, _Alloc >::assign (const _CharT * __s) [inline]
```

Set value to contents of a C string.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The C string to use. |
|------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 1128 of file `basic_string.h`.

**4.629.3.13** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( size_type __n, _CharT __c ) [inline]`

Set value to multiple characters.

**Parameters**

|                  |                                 |
|------------------|---------------------------------|
| <code>__n</code> | Length of the resulting string. |
| <code>__c</code> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 1144 of file `basic_string.h`.

**4.629.3.14** `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range `[__first, __last)`.

Definition at line 1157 of file `basic_string.h`.

**4.629.3.15** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign ( initializer_list< _CharT > __l ) [inline]`

Set value to an `initializer_list` of characters.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__l</code> | The <code>initializer_list</code> of characters to assign. |
|------------------|------------------------------------------------------------|

**Returns**

Reference to this string.

Definition at line 1167 of file basic\_string.h.

Referenced by std::basic\_string< char >::assign().

4.629.3.16 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::at ( size_type __n ) const [inline]`

Provides access to the data contained in the string.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

#### Returns

Read-only (const) reference to the character.

#### Exceptions

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 864 of file basic\_string.h.

4.629.3.17 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::at ( size_type __n ) [inline]`

Provides access to the data contained in the string.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

#### Returns

Read/write reference to the character.

#### Exceptions

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 883 of file basic\_string.h.

4.629.3.18 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 913 of file basic\_string.h.

**4.629.3.19** `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 921 of file basic\_string.h.

**4.629.3.20** `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 605 of file basic\_string.h.

Referenced by std::basic\_string< char >::crend(), std::regex\_match(), std::regex\_replace(), std::regex\_search(), and std::basic\_string< char >::rend().

**4.629.3.21** `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 616 of file basic\_string.h.

**4.629.3.22** `template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str ( ) const [inline], [noexcept]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1800 of file basic\_string.h.

Referenced by std::collate< \_CharT >::do\_compare(), std::money\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::collate< \_CharT >::do\_transform(), std::basic\_filebuf< char\_type, traits\_type >::open(), and std::operator==().

**4.629.3.23** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity ( ) const [inline], [noexcept]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 776 of file basic\_string.h.

Referenced by std::basic\_string< char >::push\_back(), and std::basic\_string< char >::shrink\_to\_fit().

**4.629.3.24** `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 680 of file basic\_string.h.

**4.629.3.25** `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 688 of file basic\_string.h.

4.629.3.26 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::clear ( ) [inline], [noexcept]`

Erases the string, making it empty.

Definition at line 803 of file basic\_string.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf()`.

4.629.3.27 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare ( const basic_string< _CharT, _Traits, _Alloc > & __str ) const [inline]`

Compare to a string.

#### Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2225 of file basic\_string.h.

Referenced by `std::sub_match< _Bi_iter >::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

4.629.3.28 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare ( size_type __pos, size_type __n, const basic_string< _CharT, _Traits, _Alloc > & __str ) const`

Compare substring to a string.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 916 of file basic\_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::min()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

4.629.3.29 `template<typename _CharT, typename _Traits, typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare ( size_type __pos1, size_type __n1, const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos2, size_type __n2 ) const`

Compare substring to a substring.

#### Parameters

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 931 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::min()`.

4.629.3.30 `template<typename _CharT, typename _Traits, typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare ( const _CharT * __s ) const`

Compare to a C string.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 949 of file `basic_string.tcc`.

References `std::min()`, and `std::size()`.

4.629.3.31 `template<typename _CharT, typename _Traits, typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare ( size_type __pos, size_type __n1, const _CharT * __s ) const`

Compare substring to a C string.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 964 of file `basic_string.tcc`.

References `std::min()`.

**4.629.3.32** `template<typename _CharT, typename _Traits, typename _Alloc > int std::basic_string<_CharT, _Traits, _Alloc>::compare( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) const`

Compare substring against a character array.

## Parameters

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

Definition at line 980 of file `basic_string.tcc`.

References `std::min()`.

**4.629.3.33** `template<typename _CharT, typename _Traits, typename _Alloc > basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::copy( _CharT * __s, size_type __n, size_type __pos = 0 ) const`

Copy substring into C string.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

## Returns

Number of characters actually copied

## Exceptions

|                          |                                     |
|--------------------------|-------------------------------------|
| <i>std::out_of_range</i> | If <code>__pos &gt; size()</code> . |
|--------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 723 of file `basic_string.tcc`.

**4.629.3.34** `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 697 of file `basic_string.h`.

**4.629.3.35** `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 706 of file `basic_string.h`.

**4.629.3.36** `template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::data( ) const [inline], [noexcept]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1810 of file `basic_string.h`.

Referenced by `std::basic_string<char>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::collate<_CharT>::do_compare()`, `std::collate<_CharT>::do_transform()`, `std::basic_string<char>::find()`, `std::basic_string<char>::find_first_not_of()`, `std::basic_string<char>::find_last_of()`, `std::match_results<_FwdIterT, _Alloc>::format()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, and `std::regex_traits<_Ch_type>::transform()`.

**4.629.3.37** `template<typename _CharT, typename _Traits, typename _Alloc> bool std::basic_string<_CharT, _Traits, _Alloc>::empty( ) const [inline], [noexcept]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 811 of file `basic_string.h`.

Referenced by `std::tr2::operator>>()`, and `std::operator>>()`.

**4.629.3.38** `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::end( ) [inline], [noexcept]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 624 of file `basic_string.h`.

Referenced by `std::basic_string<char>::crbegin()`, `std::basic_string<char>::rbegin()`, `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

4.629.339 `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 635 of file basic\_string.h.

4.629.340 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::erase ( size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

#### Parameters

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
|--------------------------------|-------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1362 of file basic\_string.h.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_string< char >::pop_back()`.

4.629.341 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::erase ( iterator __position ) [inline]`

Remove one character.

#### Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

#### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1378 of file basic\_string.h.

4.629.342 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc >::erase ( iterator __first, iterator __last )`

Remove a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

## Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 391 of file basic\_string.tcc.

**4.629.3.43** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc>::size_type std::basic_string< _CharT, _Traits, _Alloc>::find ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a C substring.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 737 of file basic\_string.tcc.

References `std::size()`.

Referenced by `std::basic_string< char >::find()`, and `std::basic_string< char >::find_first_of()`.

**4.629.3.44** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc>::find ( const basic_string< _CharT, _Traits, _Alloc> & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1846 of file basic\_string.h.

Referenced by `std::basic_string< char >::find()`.

4.629.3.45 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1861 of file `basic_string.h`.

4.629.3.46 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( _CharT __c, size_type __pos = 0 ) const [noexcept]`

Find position of a character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 760 of file `basic_string.tcc`.

References `std::size()`.

4.629.3.47 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a character not in string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2079 of file basic\_string.h.

Referenced by std::basic\_string< char >::find\_first\_not\_of().

4.629.3.48 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type  
std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of( const _CharT * __s, size_type __pos, size_type __n )  
const`

Find position of a character not in C substring.

#### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 852 of file basic\_string.tcc.

References std::size().

4.629.3.49 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc  
>::find_first_not_of( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a character not in C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2110 of file basic\_string.h.

4.629.3.50 `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type  
std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of( _CharT __c, size_type __pos = 0 ) const  
[noexcept]`

Find position of a different character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 864 of file `basic_string.tcc`.

References `std::size()`.

```
4.629.3.51 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of(const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1952 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_first_of()`.

```
4.629.3.52 template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of(const _CharT * __s, size_type __pos, size_type __n) const
```

Find position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 816 of file `basic_string.tcc`.

References `std::size()`.

```
4.629.3.53 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of(const _CharT * __s, size_type __pos = 0) const [inline]
```

Find position of a character of C string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1982 of file `basic_string.h`.

**4.629.3.54** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of ( _CharT __c, size_type __pos = 0 ) const [inline], [noexcept]`

Find position of a character.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

Definition at line 2001 of file `basic_string.h`.

**4.629.3.55** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a character not in string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2142 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

```
4.629.3.56 template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n)
const
```

Find last position of a character not in C substring.

#### Parameters

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid. |
| <code>__pos</code> | Index of character to search back from.  |
| <code>__n</code>   | Number of characters from s to consider. |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 875 of file `basic_string.tcc`.

References `std::size()`.

```
4.629.3.57 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (const _CharT * __s, size_type __pos = npos) const [inline]
```

Find last position of a character not in C string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2173 of file `basic_string.h`.

```
4.629.3.58 template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (_CharT __c, size_type __pos = npos) const
[noexcept]
```

Find last position of a different character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 896 of file `basic_string.tcc`.

References `std::size()`.

**4.629.3.59** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of ( const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2016 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_last_of()`.

**4.629.3.60** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 831 of file `basic_string.tcc`.

References `std::size()`.

**4.629.3.61** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of ( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a character of C string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2046 of file `basic_string.h`.

```
4.629.3.62 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc>::find_last_of(_CharT __c, size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

Definition at line 2065 of file `basic_string.h`.

```
4.629.3.63 template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc>::front() [inline]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 897 of file `basic_string.h`.

```
4.629.3.64 template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc>::front() const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 905 of file `basic_string.h`.

```
4.629.3.65 template<typename _CharT, typename _Traits, typename _Alloc> allocator_type std::basic_string< _CharT, _Traits, _Alloc>::get_allocator() const [inline], [noexcept]
```

Return copy of allocator used to construct this string.

Definition at line 1817 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc>::assign()`, `std::basic_string< char>::basic_string()`, `std::basic_string< _CharT, _Traits, _Alloc>::swap()`, and `std::basic_string< char>::~~basic_string()`.

4.629.3.66 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator __p, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                        |
| <code>__c</code> | The character to insert.                              |

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1185 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

4.629.3.67 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator __p, _InputIterator __beg, _InputIterator __end ) [inline]`

Insert a range of characters.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1202 of file `basic_string.h`.

4.629.3.68 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator __p, initializer_list< _CharT > __l ) [inline]`

Insert an `initializer_list` of characters.

#### Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at.      |
| <code>__l</code> | The <code>initializer_list</code> of characters to insert. |

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Definition at line 1213 of file basic\_string.h.

```
4.629.3.69 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT,
 _Traits, _Alloc>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Insert value of a string.

#### Parameters

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file basic\_string.h.

Referenced by `std::basic_string<char>::insert()`.

```
4.629.3.70 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT,
 _Traits, _Alloc>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos2,
 size_type __n) [inline]
```

Insert a substring.

#### Parameters

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert.    |
| <code>__n</code>    | Number of characters to insert.                       |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                                             |
|--------------------------------|-----------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                             |
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1255 of file basic\_string.h.

Referenced by std::basic\_string< char >::insert().

4.629.3.71 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc> & std::basic_string< _CharT, _Traits, _Alloc>::insert ( size_type __pos, const _CharT * __s, size_type __n )`

Insert a C substring.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |
| <code>__n</code>   | The number of characters to insert.                   |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 360 of file `basic_string.tcc`.

4.629.3.72 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc>::insert ( size_type __pos, const _CharT * __s ) [inline]`

Insert a C string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |

Inserts the first `n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1296 of file `basic_string.h`.

4.629.3.73 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert ( size_type __pos, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1319 of file `basic_string.h`.

4.629.3.74 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator __p, _CharT __c ) [inline]`

Insert one character.

#### Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

#### Returns

Iterator referencing newly inserted char.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1337 of file `basic_string.h`.

4.629.3.75 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::length ( ) const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 721 of file basic\_string.h.

Referenced by std::collate< \_CharT >::do\_compare(), and std::collate< \_CharT >::do\_transform().

**4.629.3.76** template<typename \_CharT, typename \_Traits, typename \_Alloc> size\_type std::basic\_string< \_CharT, \_Traits, \_Alloc >::max\_size ( ) const [inline], [noexcept]

Returns the size() of the largest possible string.

Definition at line 726 of file basic\_string.h.

Referenced by std::getline(), and std::operator>>().

**4.629.3.77** template<typename \_CharT, typename \_Traits, typename \_Alloc> basic\_string& std::basic\_string< \_CharT, \_Traits, \_Alloc >::operator+= ( const basic\_string< \_CharT, \_Traits, \_Alloc > & \_\_str ) [inline]

Append a string to this string.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

#### Returns

Reference to this string.

Definition at line 932 of file basic\_string.h.

**4.629.3.78** template<typename \_CharT, typename \_Traits, typename \_Alloc> basic\_string& std::basic\_string< \_CharT, \_Traits, \_Alloc >::operator+= ( const \_CharT \* \_\_s ) [inline]

Append a C string.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

#### Returns

Reference to this string.

Definition at line 941 of file basic\_string.h.

**4.629.3.79** template<typename \_CharT, typename \_Traits, typename \_Alloc> basic\_string& std::basic\_string< \_CharT, \_Traits, \_Alloc >::operator+= ( \_CharT \_\_c ) [inline]

Append a character.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|

#### Returns

Reference to this string.

Definition at line 950 of file basic\_string.h.

4.629.3.80 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+=( initializer_list< _CharT > __l ) [inline]`

Append an initializer\_list of characters.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__l</code> | The initializer_list of characters to be appended. |
|------------------|----------------------------------------------------|

#### Returns

Reference to this string.

Definition at line 963 of file basic\_string.h.

4.629.3.81 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator=( const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]`

Assign the value of *str* to this string.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

Definition at line 546 of file basic\_string.h.

4.629.3.82 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator=( const _CharT* __s ) [inline]`

Copy contents of *s* into this string.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>__s</code> | Source null-terminated string. |
|------------------|--------------------------------|

Definition at line 554 of file basic\_string.h.

4.629.3.83 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator=( _CharT __c ) [inline]`

Set value to string of length 1.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__c</code> | Source character. |
|------------------|-------------------|

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 565 of file basic\_string.h.

4.629.3.84 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator=( basic_string< _CharT, _Traits, _Alloc > && __str ) [inline]`

Move assign the value of *str* to this string.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 580 of file `basic_string.h`.

4.629.3.85 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= ( initializer_list< _CharT > __l ) [inline]`

Set value to string constructed from initializer list.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__l</code> | <code>std::initializer_list</code> . |
|------------------|--------------------------------------|

Definition at line 592 of file `basic_string.h`.

4.629.3.86 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

## Parameters

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

## Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 826 of file `basic_string.h`.

Referenced by `std::basic_string< char >::back()`, and `std::basic_string< char >::front()`.

4.629.3.87 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] ( size_type __pos ) [inline]`

Subscript access to the data contained in the string.

## Parameters

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

## Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 843 of file `basic_string.h`.

4.629.3.88 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::pop_back ( ) [inline]`

Remove the last character.

The string must be non-empty.

Definition at line 1407 of file basic\_string.h.

4.629.3.89 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::push_back ( _CharT __c ) [inline]`

Append a single character.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to append. |
|------------------|----------------------|

Definition at line 1052 of file basic\_string.h.

Referenced by `std::collate< _CharT >::do_transform()`, `std::basic_string< char >::operator+=(())`, `std::tr2::operator>>()`, `std::operator>>()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`.

4.629.3.90 `template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 644 of file basic\_string.h.

4.629.3.91 `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 653 of file basic\_string.h.

4.629.3.92 `template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 662 of file basic\_string.h.

4.629.3.93 `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 671 of file basic\_string.h.

4.629.3.94 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace ( size_type __pos, size_type __n, const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]`

Replace characters with value from another string.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

## Returns

Reference to this string.

## Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1429 of file `basic_string.h`.

Referenced by `std::basic_string<char>::append()`, `std::basic_string<char>::assign()`, `std::basic_string<char>::insert()`, and `std::basic_string<char>::replace()`.

**4.629.3.95** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos2, size_type __n2 ) [inline]`

Replace characters with value from another string.

## Parameters

|                     |                                                      |
|---------------------|------------------------------------------------------|
| <code>__pos1</code> | Index of first character to replace.                 |
| <code>__n1</code>   | Number of characters to be replaced.                 |
| <code>__str</code>  | String to insert.                                    |
| <code>__pos2</code> | Index of first character of <code>str</code> to use. |
| <code>__n2</code>   | Number of characters from <code>str</code> to use.   |

## Returns

Reference to this string.

## Exceptions

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1451 of file `basic_string.h`.

Referenced by `std::basic_string<char>::replace()`.

4.629.3.96 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::replace ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 )`

Replace characters with value of a C substring.

#### Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.             |
| <code>__n1</code>  | Number of characters to be replaced.             |
| <code>__s</code>   | C string to insert.                              |
| <code>__n2</code>  | Number of characters from <code>s</code> to use. |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> .              |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 414 of file `basic_string.tcc`.

4.629.3.97 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( size_type __pos, size_type __n1, const _CharT * __s ) [inline]`

Replace characters with value of a C string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1496 of file `basic_string.h`.

4.629.3.98 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c ) [inline]`

Replace characters with multiple characters.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1520 of file `basic_string.h`.

4.629.3.99 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace ( iterator __i1, iterator __i2, const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]`

Replace range of characters with string.

#### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

4.629.3.100 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, const _CharT* __s, size_type __n ) [inline]`

Replace range of characters with C substring.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1557 of file `basic_string.h`.

4.629.3.101 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, const _CharT* __s ) [inline]`

Replace range of characters with C string.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1578 of file `basic_string.h`.

4.629.3.102 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, size_type __n, _CharT __c ) [inline]`

Replace range of characters with multiple characters.

## Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__n</code>  | Number of characters to insert.                 |
| <code>__c</code>  | Character to insert.                            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1599 of file `basic_string.h`.

4.629.3.103 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2 ) [inline]`

Replace range of characters with range.

## Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1623 of file `basic_string.h`.

4.629.3.104 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace ( iterator __i1, iterator __i2, initializer_list<_CharT> __l ) [inline]`

Replace range of characters with `initializer_list`.

## Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1692 of file `basic_string.h`.

Referenced by `std::basic_string<char>::replace()`.

**4.629.3.105** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::reserve ( size_type __res_arg = 0 )`

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                          |                                                             |
|--------------------------|-------------------------------------------------------------|
| <i>std::length_error</i> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 502 of file `basic_string.tcc`.

References `std::size()`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::tr2::operator>>()`, `std::operator>>()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_string<char>::push_back()`, and `std::basic_string<char>::shrink_to_fit()`.

**4.629.3.106** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::resize ( size_type __n, _CharT __c )`

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
| <code>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 640 of file `basic_string.tcc`.

References `std::size()`.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get().

**4.629.3.107** `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::resize( size_type __n ) [inline]`

Resizes the string to the specified number of characters.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 753 of file basic\_string.h.

Referenced by std::basic\_string< char >::resize().

**4.629.3.108** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = npos ) const [inline], [noexcept]`

Find last position of a string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1891 of file basic\_string.h.

Referenced by std::basic\_string< char >::find\_last\_of(), and std::basic\_string< char >::rfind().

**4.629.3.109** `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

#### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string to locate.                        |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 778 of file basic\_string.tcc.

References std::min(), and std::size().

4.629.3.110 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a C string.

#### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1921 of file basic\_string.h.

4.629.3.111 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind( _CharT __c, size_type __pos = npos ) const [noexcept]`

Find last position of a character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 799 of file basic\_string.tcc.

References std::size().

4.629.3.112 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit( ) [inline]`

A non-binding request to reduce capacity() to size().

Definition at line 759 of file basic\_string.h.

4.629.3.113 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::size( ) const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 715 of file basic\_string.h.

Referenced by std::basic\_string< \_CharT, \_Traits, \_Alloc >::append(), std::basic\_string< char >::assign(), std::basic\_string< char >::at(), std::basic\_string< char >::back(), std::basic\_string< char >::cend(), std::basic\_string< char

>::clear(), std::basic\_string< char >::compare(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::compare(), std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::dynamic\_bitset(), std::basic\_string< char >::empty(), std::basic\_string< char >::end(), std::basic\_string< char >::find(), std::basic\_string< char >::find\_first\_not\_of(), std::basic\_string< char >::find\_last\_of(), std::match\_results< \_FwdIterT, \_Alloc >::format(), std::basic\_string< char >::insert(), std::operator+(), std::tr2::operator>>(), std::basic\_string< char >::operator[](), std::basic\_string< char >::pop\_back(), std::basic\_string< char >::push\_back(), std::basic\_string< char >::replace(), std::basic\_string< char >::shrink\_to\_fit(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), and std::regex\_traits< \_Ch\_type >::transform().

**4.629.3.114** template<typename \_CharT, typename \_Traits, typename \_Alloc> **basic\_string** std::basic\_string< \_CharT, \_Traits, \_Alloc >::substr ( size\_type \_\_pos = 0, size\_type \_\_n = npos ) const [inline]

Get a substring.

#### Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

#### Returns

The new string.

#### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2206 of file `basic_string.h`.

**4.629.3.115** template<typename \_CharT, typename \_Traits, typename \_Alloc> void std::basic\_string< \_CharT, \_Traits, \_Alloc >::swap ( basic\_string< \_CharT, \_Traits, \_Alloc > & \_\_s )

Swap contents with another string.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 519 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`.

Referenced by `std::basic_string< char >::assign()`, `std::basic_string< char >::operator=()`, and `std::swap()`.

### 4.629.4 Member Data Documentation

**4.629.4.1** template<typename \_CharT, typename \_Traits, typename \_Alloc> const basic\_string< \_CharT, \_Traits, \_Alloc >::size\_type std::basic\_string< \_CharT, \_Traits, \_Alloc >::npos [static]

Value returned by various member functions when they fail.

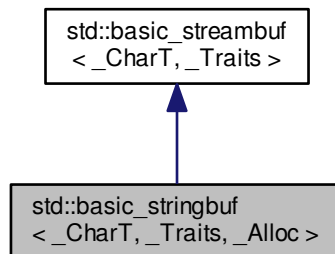
Definition at line 285 of file `basic_string.h`.

The documentation for this class was generated from the following files:

- [basic\\_string.h](#)
- [basic\\_string.tcc](#)

#### 4.630 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

Inheritance diagram for std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >:



#### Public Types

- typedef \_\_string\_type::size\_type **\_\_size\_type**
- typedef [basic\\_streambuf](#) < char\_type, traits\_type > **\_\_streambuf\_type**
- typedef [basic\\_string](#) < char\_type, \_Traits, \_Alloc > **\_\_string\_type**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

#### Public Member Functions

- [basic\\_stringbuf](#) (ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [basic\\_stringbuf](#) (const \_\_string\_type &\_\_str, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [locale](#) getloc () const
- [streamsize](#) in\_avail ()
- [locale](#) pubimbue (const [locale](#) &\_\_loc)
- int\_type sbumpc ()
- int\_type sgetc ()
- [streamsize](#) sgetn (char\_type \*\_\_s, [streamsize](#) \_\_n)
- int\_type snextc ()

- int\_type [sputbackc](#) (char\_type \_\_c)
- int\_type [sputc](#) (char\_type \_\_c)
- streamsize [sputn](#) (const char\_type \* \_\_s, streamsize \_\_n)
- \_\_string\_type [str](#) () const
- void [str](#) (const \_\_string\_type & \_\_s)
- int\_type [sungetc](#) ()
- basic\_streambuf \* [pubsetbuf](#) (char\_type \* \_\_s, streamsize \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int [pubsync](#) ()

#### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) (streamsize \_\_n)
- void [\\_\\_safe\\_pbump](#) (streamsize \_\_n)
- void [\\_M\\_pbump](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend, off\_type \_\_off)
- void [\\_M\\_stringbuf\\_init](#) (ios\_base::openmode \_\_mode)
- void [\\_M\\_sync](#) (char\_type \* \_\_base, \_\_size\_type \_\_i, \_\_size\_type \_\_o)
- void [\\_M\\_update\\_egptr](#) ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const locale & \_\_loc)
- virtual int\_type [overflow](#) (int\_type \_\_c=traits\_type::eof())
- virtual int\_type [pbackfail](#) (int\_type \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual pos\_type [seekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- virtual pos\_type [seekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- virtual \_\_streambuf\_type \* [setbuf](#) (char\_type \* \_\_s, streamsize \_\_n)
- void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual streamsize [showmanyc](#) ()
- virtual int [sync](#) ()
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- virtual streamsize [xsgetn](#) (char\_type \* \_\_s, streamsize \_\_n)
- virtual streamsize [xsputn](#) (const char\_type \* \_\_s, streamsize \_\_n)
- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const
- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

## Protected Attributes

- [locale \\_M\\_buf\\_locale](#)
- [char\\_type \\* \\_M\\_in\\_beg](#)
- [char\\_type \\* \\_M\\_in\\_cur](#)
- [char\\_type \\* \\_M\\_in\\_end](#)
- [ios\\_base::openmode \\_M\\_mode](#)
- [char\\_type \\* \\_M\\_out\\_beg](#)
- [char\\_type \\* \\_M\\_out\\_cur](#)
- [char\\_type \\* \\_M\\_out\\_end](#)
- [\\_\\_string\\_type \\_M\\_string](#)

## 4.630.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_stringbuf< _CharT, _Traits, _Alloc >
```

The actual work of input and output (for std::string).

## Template Parameters

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                          |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> . |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <a href="#">allocator&lt;_CharT&gt;</a> .              |

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic\\_string](#). (Paraphrased from [27.7.1]/1.)

For this class, open modes (of type [ios\\_base::openmode](#)) have [in](#) set if the input sequence can be read, and [out](#) set if the output sequence can be written.

Definition at line 64 of file [sstream](#).

## 4.630.2 Constructor &amp; Destructor Documentation

4.630.2.1 

```
template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Starts with an empty string buffer.

## Parameters

|                        |                                                 |
|------------------------|-------------------------------------------------|
| <a href="#">__mode</a> | Whether the buffer can read, or write, or both. |
|------------------------|-------------------------------------------------|

The default constructor initializes the parent class using its own default ctor.

Definition at line 98 of file [sstream](#).

4.630.2.2 

```
template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (const __string_type & __str, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

This constructor initializes the parent class using its own default ctor.

Definition at line 111 of file sstream.

## 4.630.3 Member Function Documentation

4.630.3.1 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eback ( )`  
`const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, and `std::basic_streambuf< char_type, traits_type >::sungetc()`.

4.630.3.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( )`  
`const [inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

4.630.3.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eptr ( )`  
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence

- `eptr()` returns the end pointer for the output sequence

Definition at line 535 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputc()`.

**4.630.3.4** `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::gbump ( int __n )`  
`[inline], [protected], [inherited]`

Moving the read position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Definition at line 498 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, and `std::basic_streambuf< char_type, traits_type >::uflow()`.

**4.630.3.5** `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const`  
`[inline], [inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**4.630.3.6** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const`  
`[inline], [protected], [inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 485 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, and `std::basic_streambuf< char_type, traits_type >::uflow()`.

4.630.3.7 `template<typename _CharT, typename _Traits> virtual void std::basic_streambuf< _CharT, _Traits >::imbue ( const locale & __loc ) [inline], [protected], [virtual], [inherited]`

Changes translations.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 576 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

4.630.3.8 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline], [inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.630.3.9 `template<class _CharT, class _Traits, class _Alloc> basic_stringbuf< _CharT, _Traits, _Alloc >::int_type basic_stringbuf< _CharT, _Traits, _Alloc >::overflow ( int_type __c = traits_type::eof() ) [protected], [virtual]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

#### Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns eof().

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 80 of file sstream.tcc.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::assign\(\)](#), [std::max\(\)](#), [std::min\(\)](#), [std::ios\\_base::out](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::push\\_back\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::reserve\(\)](#).

```
4.630.3.10 template<class _CharT, class _Traits, class _Alloc > basic_stringbuf< _CharT, _Traits, _Alloc >::int_type
 basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail (int_type __c = traits_type::eof())
 [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

eof() on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns eof().

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 46 of file sstream.tcc.

References [std::ios\\_base::out](#).

```
4.630.3.11 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pbase ()
 const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::str\(\)](#).

4.630.3.12 `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::pbump ( int __n )`  
`[inline], [protected], [inherited]`

Moving the write position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputc()`.

4.630.3.13 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( )`  
`const [inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputc()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

4.630.3.14 `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale & __loc )`  
`[inline], [inherited]`

Entry point for imbue().

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

#### Returns

The previous locale.

Calls the derived `imbue(__loc)`.

Definition at line 209 of file streambuf.

4.630.3.15 `template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff ( off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out )`  
`[inline], [inherited]`

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

```
4.630.3.16 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos
(pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline],
[inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

```
4.630.3.17 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits
>::pubsetbuf(char_type * __s, streamsize __n) [inline],[inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

```
4.630.3.18 template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ()
[inline],[inherited]
```

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

```
4.630.3.19 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()
[inline],[inherited]
```

Getting the next character.

## Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `ufLOW()`.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream_iterator< _CharT, _Traits >::operator++()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

```
4.630.3.20 template<class _CharT, class _Traits, class _Alloc > basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type
 basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(off_type, ios_base::seekdir, ios_base::openmode =
 ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 149 of file `sstream.tcc`.

References `std::ios_base::cur`, `std::ios_base::end`, `std::ios_base::in`, and `std::ios_base::out`.

```
4.630.3.21 template<class _CharT, class _Traits, class _Alloc > basic_stringbuf< _CharT, _Traits, _Alloc
 >::pos_type basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(pos_type, ios_base::openmode =
 ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 197 of file `sstream.tcc`.

References `std::ios_base::in`, and `std::ios_base::out`.

```
4.630.3.22 template<typename _CharT, typename _Traits, typename _Alloc> virtual __streambuf_type*
 std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf(char_type * __s, streamsize __n) [inline],
 [protected], [virtual]
```

Manipulates the buffer.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__s</code> | Pointer to a buffer area.  |
| <code>__n</code> | Size of <code>__s</code> . |

#### Returns

`this`

If no buffer has already been created, and both `__s` and `__n` are non-zero, then `__s` is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 202 of file `sstream`.

References `std::basic_string< _CharT, _Traits, _Alloc >::clear()`.

4.630.3.23 `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setg ( char_type * __gbeg, char_type * __gnext, char_type * __gend )` [inline], [protected], [inherited]

Setting the three read area pointers.

#### Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

#### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 509 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

4.630.3.24 `template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setp ( char_type * __pbeg, char_type * __pend )` [inline], [protected], [inherited]

Setting the three write area pointers.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

#### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 555 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

4.630.3.25 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( )` [inline], [inherited]

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_streambuf< char_type, traits_type >::sngetc()`.

4.630.3.26 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s, streamsize __n) [inline], [inherited]`

Entry point for xsgetn.

#### Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns xsgetn(\_\_s,\_\_n). The effect is to fill \_\_s[0] through \_\_s[\_\_n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.630.3.27 `template<typename _CharT, typename _Traits, typename _Alloc> virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc ( ) [inline], [protected], [virtual]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 170 of file sstream.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::M_mode`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::in`.

4.630.3.28 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline], [inherited]`

Getting the next character.

#### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.630.3.29 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type __c) [inline], [inherited]`

Pushing characters back into the input stream.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

#### Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 372 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

4.630.3.30 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c) [inline], [inherited]`

Entry point for all single-character output functions.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

#### Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(-__c)`.

Definition at line 424 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

4.630.3.31 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * __s, streamsize __n) [inline], [inherited]`

Entry point for all single-character output functions.

#### Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 450 of file `streambuf`.

4.630.3.32 `template<typename _CharT, typename _Traits, typename _Alloc> __string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

A copy of one of the underlying sequences.

*If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1*

Definition at line 126 of file sstream.

References `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

4.630.3.33 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( const __string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Deallocates any previous stored sequence, then copies `s` to use as a new one.

Definition at line 150 of file sstream.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::M_mode`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

4.630.3.34 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline], [inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

4.630.3.35 `template<typename _CharT, typename _Traits> virtual int std::basic_streambuf< _CharT, _Traits >::sync ( void ) [inline], [protected], [virtual], [inherited]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 627 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubsync\(\)](#).

**4.630.3.36** `template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( )`  
`[inline], [protected], [virtual], [inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 700 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#).

**4.630.3.37** `template<class _CharT, class _Traits, class _Alloc> basic_stringbuf< _CharT, _Traits, _Alloc >::int_type`  
`basic_stringbuf< _CharT, _Traits, _Alloc >::underflow ( )` `[protected], [virtual]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

**Note**

Base class version does nothing, returns eof().

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 131 of file sstream.tcc.

References [std::ios\\_base::in](#).

**4.630.3.38** `template<typename _CharT, typename _Traits> streamsize basic_streambuf< _CharT, _Traits >::xsgetn (`  
`char_type * __s, streamsize __n )` `[protected], [virtual], [inherited]`

Multiple character extraction.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 46 of file `streambuf.tcc`.

References `std::min()`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sgetn()`.

**4.630.3.39** `template<typename _CharT, typename _Traits> streamsize basic_streambuf< _CharT, _Traits >::xsputn ( const char_type * __s, streamsize __n )` `[protected]`, `[virtual]`, `[inherited]`

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 80 of file `streambuf.tcc`.

References `std::min()`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sputn()`.

**4.630.4 Member Data Documentation**

**4.630.4.1** `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::M_buf_locale` `[protected]`, `[inherited]`

Current locale setting.

Definition at line 192 of file `streambuf`.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::basic\_filebuf(), std::basic\_streambuf< char\_type, traits\_type >::getloc(), and std::basic\_streambuf< char\_type, traits\_type >::pubimbue().

**4.630.4.2** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::M\_in\_beg  
[protected], [inherited]

Start of get area.

Definition at line 184 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::eback(), and std::basic\_streambuf< char\_type, traits\_type >::setg().

**4.630.4.3** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::M\_in\_cur  
[protected], [inherited]

Current read area.

Definition at line 185 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::gbump(), std::basic\_streambuf< char\_type, traits\_type >::gptr(), and std::basic\_streambuf< char\_type, traits\_type >::setg().

**4.630.4.4** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::M\_in\_end  
[protected], [inherited]

End of get area.

Definition at line 186 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::egptr(), and std::basic\_streambuf< char\_type, traits\_type >::setg().

**4.630.4.5** template<typename \_CharT, typename \_Traits, typename \_Alloc> ios\_base::openmode std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::M\_mode [protected]

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 83 of file sstream.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str().

**4.630.4.6** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::M\_out\_beg  
[protected], [inherited]

Start of put area.

Definition at line 187 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::pbase(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

**4.630.4.7** template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::M\_out\_cur  
[protected], [inherited]

Current put area.

Definition at line 188 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::pbump(), std::basic\_streambuf< char\_type, traits\_type >::pptr(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

4.630.4.8 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::M_out_end`  
`[protected], [inherited]`

End of put area.

Definition at line 189 of file streambuf.

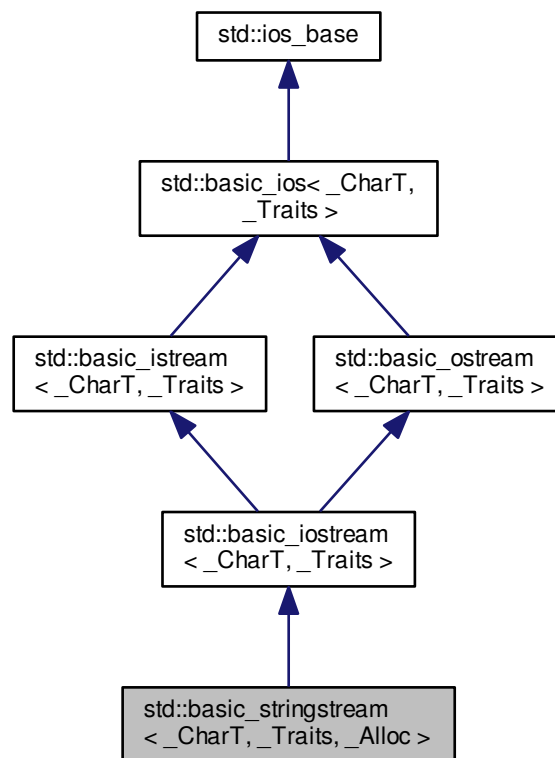
Referenced by `std::basic_streambuf<char_type, traits_type>::epptr()`, and `std::basic_streambuf<char_type, traits_type>::setp()`.

The documentation for this class was generated from the following files:

- [sstream](#)
- [sstream.tcc](#)

## 4.631 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference

Inheritance diagram for `std::basic_stringstream<_CharT, _Traits, _Alloc>`:



### Public Types

- typedef `ctype<_CharT> __ctype_type`

- typedef [ctype](#)<\_CharT> **\_\_ctype\_type**
- typedef [basic\\_ios](#)<\_CharT, \_Traits> **\_\_ios\_type**
- typedef [basic\\_ios](#)<\_CharT, \_Traits> **\_\_ios\_type**
- typedef [basic\\_istream](#)<char\_type, traits\_type> **\_\_istream\_type**
- typedef [basic\\_istream](#)<\_CharT, \_Traits> **\_\_istream\_type**
- typedef [num\\_get](#)<\_CharT, istreambuf\_iterator<\_CharT, \_Traits>> **\_\_num\_get\_type**
- typedef [num\\_put](#)<\_CharT, ostreambuf\_iterator<\_CharT, \_Traits>> **\_\_num\_put\_type**
- typedef [basic\\_ostream](#)<\_CharT, \_Traits> **\_\_ostream\_type**
- typedef [basic\\_streambuf](#)<\_CharT, \_Traits> **\_\_streambuf\_type**
- typedef [basic\\_streambuf](#)<\_CharT, \_Traits> **\_\_streambuf\_type**
- typedef [basic\\_string](#)<\_CharT, \_Traits, \_Alloc> **\_\_string\_type**
- typedef [basic\\_stringbuf](#)<\_CharT, \_Traits, \_Alloc> **\_\_stringbuf\_type**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
- typedef void(\* [event\\_callback](#))([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags **fmtflags**
- typedef traits\_type::int\_type **int\_type**
- typedef int **io\_state**
- typedef \_ios\_istate **istate**
- typedef traits\_type::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_ios\_Openmode **openmode**
- typedef traits\_type::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_ios\_Seekdir **seekdir**
- typedef [std::streamoff](#) **streamoff**
- typedef [std::streampos](#) **streampos**
- typedef \_Traits **traits\_type**
  
- typedef [num\\_put](#)<\_CharT, ostreambuf\_iterator<\_CharT, \_Traits>> **\_\_num\_put\_type**

## Public Member Functions

- `basic_stringstream` (`ios_base::openmode` \_\_m=`ios_base::out`|`ios_base::in`)
- `basic_stringstream` (`const` \_\_string\_type &\_\_str, `ios_base::openmode` \_\_m=`ios_base::out`|`ios_base::in`)
- `~basic_stringstream` ()
- `const` `locale` & `_M_getloc` () `const`
- `void` `_M_setstate` (`iosstate` \_\_state)
- `bool` `bad` () `const`
- `void` `clear` (`iosstate` \_\_state=`goodbit`)
- `basic_ios` & `copyfmt` (`const` `basic_ios` &\_\_rhs)
- `bool` `eof` () `const`
- `iosstate` `exceptions` () `const`
- `void` `exceptions` (`iosstate` \_\_except)
- `bool` `fail` () `const`
- `char_type` `fill` () `const`
- `char_type` `fill` (`char_type` \_\_ch)
- `fmtflags` `flags` () `const`
- `fmtflags` `flags` (`fmtflags` \_\_fmtfl)
- \_\_ostream\_type & `flush` ()
- `streamsize` `gcount` () `const`
- `locale` `getloc` () `const`
- `bool` `good` () `const`
- `locale` `imbue` (`const` `locale` &\_\_loc)
- `long` & `word` (`int` \_\_ix)
- `char` `narrow` (`char_type` \_\_c, `char` \_\_default) `const`
- \_\_ostream\_type & `operator<<` (`const` `void` \*\_\_p)
- \_\_ostream\_type & `operator<<` (\_\_streambuf\_type \*\_\_sb)
- \_\_istream\_type & `operator>>` (`void` \*&\_\_p)
- \_\_istream\_type & `operator>>` (\_\_streambuf\_type \*\_\_sb)
- `streamsize` `precision` () `const`
- `streamsize` `precision` (`streamsize` \_\_prec)
- `void` \*& `pword` (`int` \_\_ix)
- `basic_streambuf<_CharT, _Traits>` \* `rdbuf` (`basic_streambuf<_CharT, _Traits>` \*\_\_sb)
- \_\_stringbuf\_type \* `rdbuf` () `const`
- `iosstate` `rdstate` () `const`
- `void` `register_callback` (`event_callback` \_\_fn, `int` \_\_index)
- \_\_ostream\_type & `seekp` (`pos_type`)
- \_\_ostream\_type & `seekp` (`off_type`, `ios_base::seekdir`)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
- `void` `setstate` (`iosstate` \_\_state)
- \_\_string\_type `str` () `const`
- `void` `str` (`const` \_\_string\_type &\_\_s)
- `pos_type` `tellp` ()
- `basic_ostream<_CharT, _Traits>` \* `tie` () `const`
- `basic_ostream<_CharT, _Traits>` \* `tie` (`basic_ostream<_CharT, _Traits>` \*\_\_tiestr)
- `void` `unsetf` (`fmtflags` \_\_mask)
- `char_type` `widen` (`char` \_\_c) `const`
- `streamsize` `width` () `const`
- `streamsize` `width` (`streamsize` \_\_wide)

- [\\_\\_istream\\_type & operator>> \(\\_\\_istream\\_type &\(\\_\\_pf\)\(\\_\\_istream\\_type &\)\)](#)
- [\\_\\_istream\\_type & operator>> \(\\_\\_ios\\_type &\(\\_\\_pf\)\(\\_\\_ios\\_type &\)\)](#)
- [\\_\\_istream\\_type & operator>> \(ios\\_base &\(\\_\\_pf\)\(ios\\_base &\)\)](#)

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_istream\\_type & operator>> \(bool &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(short &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(unsigned short &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(int &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(unsigned int &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(long &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(unsigned long &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(long long &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(unsigned long long &\\_\\_n\)](#)
- [\\_\\_istream\\_type & operator>> \(float &\\_\\_f\)](#)
- [\\_\\_istream\\_type & operator>> \(double &\\_\\_f\)](#)
- [\\_\\_istream\\_type & operator>> \(long double &\\_\\_f\)](#)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [int\\_type get \(\)](#)
- [\\_\\_istream\\_type & get \(char\\_type &\\_\\_c\)](#)
- [\\_\\_istream\\_type & get \(char\\_type \\*\\_\\_s, streamsize \\_\\_n, char\\_type \\_\\_delim\)](#)
- [\\_\\_istream\\_type & get \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
- [\\_\\_istream\\_type & get \(\\_\\_streambuf\\_type &\\_\\_sb, char\\_type \\_\\_delim\)](#)
- [\\_\\_istream\\_type & get \(\\_\\_streambuf\\_type &\\_\\_sb\)](#)
- [\\_\\_istream\\_type & getline \(char\\_type \\*\\_\\_s, streamsize \\_\\_n, char\\_type \\_\\_delim\)](#)
- [\\_\\_istream\\_type & getline \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
- [\\_\\_istream\\_type & ignore \(streamsize \\_\\_n, int\\_type \\_\\_delim\)](#)
- [\\_\\_istream\\_type & ignore \(streamsize \\_\\_n\)](#)
- [\\_\\_istream\\_type & ignore \(\)](#)
- [int\\_type peek \(\)](#)
- [\\_\\_istream\\_type & read \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
- [streamsize readsome \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
- [\\_\\_istream\\_type & putback \(char\\_type \\_\\_c\)](#)
- [\\_\\_istream\\_type & unget \(\)](#)
- [int sync \(\)](#)
- [pos\\_type tellg \(\)](#)
- [\\_\\_istream\\_type & seekg \(pos\\_type\)](#)

- `__istream_type` & `seekg` (`off_type`, `ios_base::seekdir`)
- `operator void *` () const
- `bool operator!` () const
- `__ostream_type` & `operator<<` (`__ostream_type` &(\*\_\_pf)(`__ostream_type` &))
- `__ostream_type` & `operator<<` (`__ios_type` &(\*\_\_pf)(`__ios_type` &))
- `__ostream_type` & `operator<<` (`ios_base` &(\*\_\_pf)(`ios_base` &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type` & `operator<<` (`long __n`)
- `__ostream_type` & `operator<<` (`unsigned long __n`)
- `__ostream_type` & `operator<<` (`bool __n`)
- `__ostream_type` & `operator<<` (`short __n`)
- `__ostream_type` & `operator<<` (`unsigned short __n`)
- `__ostream_type` & `operator<<` (`int __n`)
- `__ostream_type` & `operator<<` (`unsigned int __n`)
- `__ostream_type` & `operator<<` (`long long __n`)
- `__ostream_type` & `operator<<` (`unsigned long long __n`)
- `__ostream_type` & `operator<<` (`double __f`)
- `__ostream_type` & `operator<<` (`float __f`)
- `__ostream_type` & `operator<<` (`long double __f`)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type` & `put` (`char_type __c`)
- `void _M_write` (`const char_type *__s`, `streamsize __n`)
- `__ostream_type` & `write` (`const char_type *__s`, `streamsize __n`)

### Static Public Member Functions

- static `bool sync_with_stdio` (`bool __sync=true`)
- static `int xalloc` () throw ()

## Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  `__istream_type` & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT >  
  `__ostream_type` & `_M_insert` (\_ValueT \_\_v)
- void `init` (`basic_streambuf`<\_CharT, \_Traits> \*\_\_sb)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 4.631.1 Detailed Description

template<typename \_CharT, typename \_Traits, typename \_Alloc> class std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>

Controlling input and output for std::string.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 502 of file `sstream`.

## 4.631.2 Member Typedef Documentation

4.631.2.1 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type` [inherited]

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.631.2.2 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i)` [inherited]

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

#### 4.631.2.3 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 255 of file `ios_base.h`.

#### 4.631.2.4 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 330 of file `ios_base.h`.

#### 4.631.2.5 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 361 of file `ios_base.h`.

#### 4.631.2.6 `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

### 4.631.3 Member Enumeration Documentation

#### 4.631.3.1 `enum std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

## 4.631.4 Constructor &amp; Destructor Documentation

4.631.4.1 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream( ios_base::openmode __m = ios_base::out | ios_base::in ) [inline], [explicit]`

Default constructor starts with an empty string buffer.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__m</code> | Whether the buffer can read, or write, or both. |
|------------------|-------------------------------------------------|

Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 537 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

4.631.4.2 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream( const __string_type &__str, ios_base::openmode __m = ios_base::out | ios_base::in ) [inline], [explicit]`

Starts with an existing string buffer.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__str</code> | A string to copy as a starting buffer.          |
| <code>__m</code>   | Whether the buffer can read, or write, or both. |

Initializes `sb` using `__str` and `__m`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 553 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

4.631.4.3 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc>::~~basic_stringstream( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 564 of file `sstream`.

## 4.631.5 Member Function Documentation

4.631.5.1 `const locale& std::ios_base::M_getloc( ) const [inline], [inherited]`

Locale access.

## Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::time\_put< \_CharT, \_Outlter >::put().

**4.631.5.2** template<typename \_CharT, typename \_Traits> void std::basic\_ostream< \_CharT, \_Traits >::M.write ( const char\_type \* \_\_s, streamsize \_\_n ) [inline], [inherited]

Core write functionality, without sentry.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 311 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

**4.631.5.3** template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::bad ( ) const [inline], [inherited]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic\_ios.h.

**4.631.5.4** template<typename \_CharT, typename \_Traits > void basic\_ios< \_CharT, \_Traits >::clear ( iostate \_\_state = goodbit ) [inherited]

[Re]sets the error state.

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.631.5.5** template<typename \_CharT, typename \_Traits > basic\_ios< \_CharT, \_Traits > & basic\_ios< \_CharT, \_Traits >::copyfmt ( const basic\_ios< \_CharT, \_Traits > & \_\_rhs ) [inherited]

Copies fields of \_\_rhs into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

## Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, `std::tie()`, and `std::ios_base::width()`.

**4.631.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const` `[inline]`, `[inherited]`

Fast error checking.

## Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 184 of file `basic_ios.h`.

**4.631.5.7** `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const` `[inline]`, `[inherited]`

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**4.631.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except )` `[inline]`, `[inherited]`

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (
 __gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic\_ios.h.

**4.631.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const` `[inline]`, `[inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::operator void \*(), std::basic\_ios<char, \_Traits>::operator!(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_istream<\_CharT, \_Traits>::tellg(), std::basic\_ostream<\_CharT, \_Traits>::tellp(), and std::regex\_traits<\_Ch\_type>::value().

**4.631.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill ( ) const` `[inline]`, `[inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 364 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::basic\_ios<char, \_Traits>::fill().

**4.631.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill ( char_type __ch )` `[inline]`, `[inherited]`

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 384 of file basic\_ios.h.

**4.631.5.12 fmtflags std::ios\_base::flags ( ) const [inline],[inherited]**

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 551 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.631.5.13 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline],[inherited]**

Setting new format flags all at once.

**Parameters**

|                |                       |
|----------------|-----------------------|
| <u>__fmtfl</u> | The new flags to set. |
|----------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with \_\_fmtfl.

Definition at line 562 of file ios\_base.h.

**4.631.5.14 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & basic\_ostream< \_CharT, \_Traits >::flush ( ) [inherited]**

Synchronizing the stream buffer.

**Returns**

\*this

If rdbuf ( ) is a null pointer, changes nothing.

Otherwise, calls rdbuf ( )->pubsync ( ) , and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

Referenced by std::flush().

4.631.5.15 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( )`  
`const [inline],[inherited]`

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.631.5.16 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.631.5.17 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

#### Returns

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

4.631.5.18 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**4.631.5.19** template<typename \_CharT, typename \_Traits> \_\_istream\_type& std::basic\_istream< \_CharT, \_Traits >::get (char\_type \* \_\_s, streamsize \_\_n) [inline], [inherited]

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

## Returns

\*this

Returns `get(__s, __n, widen("\n"))`.

Definition at line 354 of file istream.

**4.631.5.20** template<typename \_CharT, typename \_Traits> basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::get ( \_\_streambuf\_type & \_\_sb, char\_type \_\_delim ) [inherited]

Extraction into another streambuf.

## Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

## Returns

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::putc()`.

4.631.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline], [inherited]`

Extraction into another streambuf.

## Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

## Returns

\*this

Returns `get(__sb, widen("\n"))`.

Definition at line 387 of file istream.

4.631.5.22 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::sngetc()`.

**4.631.5.23** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline], [inherited]`

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s, __n, widen("\n"))`.

Definition at line 427 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**4.631.5.24** `locale std::ios_base::getloc ( ) const [inline], [inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::ws()`.

4.631.5.25 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const`  
`[inline], [inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

4.631.5.26 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim )` `[inherited]`

Discarding characters.

#### Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

#### Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snxctc()`.

4.631.5.27 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore ( streamsize __n )` `[inherited]`

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), std::basic\_ios<\_CharT, \_Traits>::setstate(), std::basic\_streambuf<\_CharT, \_Traits>::sgetc(), and std::basic\_streambuf<\_CharT, \_Traits>::snextc().

**4.631.5.28** template<typename \_CharT, typename \_Traits> **basic\_istream**<\_CharT, \_Traits> & **basic\_istream**<\_CharT, \_Traits>::ignore( void ) [inherited]

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), std::basic\_streambuf<\_CharT, \_Traits>::sbumpc(), and std::basic\_ios<\_CharT, \_Traits>::setstate().

**4.631.5.29** template<typename \_CharT, typename \_Traits> **locale basic\_ios**<\_CharT, \_Traits>::imbue( const locale & \_\_loc ) [inherited]

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls ios\_base::imbue(loc), and if a stream buffer is associated with this stream, calls that buffer's pubimbue(loc).

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file basic\_ios.tcc.

References std::ios\_base::imbue().

Referenced by std::operator<<().

**4.631.5.30** template<typename \_CharT, typename \_Traits> void **basic\_ios**<\_CharT, \_Traits>::init( **basic\_streambuf**<\_CharT, \_Traits> \* \_\_sb ) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_ios< char, \_Traits >::basic\_ios(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >::basic\_ostreamstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

**4.631.5.31** long& std::ios\_base::iword ( int \_\_ix ) [inline], [inherited]

Access to integer array.

#### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

#### Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios\_base.h.

**4.631.5.32** template<typename \_CharT, typename \_Traits> char std::basic\_ios< \_CharT, \_Traits >::narrow ( char\_type \_\_c, char \_\_dfault ) const [inline], [inherited]

Squeezes characters.

#### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 424 of file basic\_ios.h.

**4.631.5.33** template<typename \_CharT, typename \_Traits> std::basic\_ios< \_CharT, \_Traits >::operator void \* ( ) const [inline], [inherited]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic\_ios.h.

4.631.5.34 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline], [inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file `basic_ios.h`.

4.631.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ostream_type &(*)(__ostream_type &)__pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

4.631.5.36 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type &(*)(__ios_type &)__pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

4.631.5.37 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( ios_base &(*)(ios_base &)__pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

4.631.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( long __n ) [inline], [inherited]`

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

4.631.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned long __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.631.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( bool __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.631.5.41 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::operator<<( short __n ) [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

4.631.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( unsigned short __n ) [inline], [inherited]`

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

**4.631.5.43** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**4.631.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline], [inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

**4.631.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long long __n ) [inline], [inherited]`

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

```
4.631.5.46 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(unsigned long long __n) [inline],[inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

```
4.631.5.47 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(double __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

```
4.631.5.48 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(float __f) [inline],[inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

```
4.631.5.49 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(long double __f) [inline],[inherited]
```

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.631.5.50 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( const void * __p ) [inline], [inherited]`

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.631.5.51 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.631.5.52 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( __istream_type &(*)(__istream_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file `istream`.

4.631.5.53 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( __ios_type &(*)(__ios_type &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 124 of file `istream`.

4.631.5.54 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( ios_base &(*)(ios_base &) __pf ) [inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

4.631.5.55 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( bool & __n ) [inline], [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

4.631.5.56 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::operator>>( short & __n ) [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.631.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned short & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.631.5.58** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.631.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned int & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 182 of file istream.

4.631.5.60 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 186 of file istream.

4.631.5.61 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( unsigned long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 190 of file istream.

4.631.5.62 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( long long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 195 of file istream.

4.631.5.63 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline], [inherited]`

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

4.631.5.64 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( float & __f ) [inline], [inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

4.631.5.65 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( double & __f ) [inline], [inherited]`

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

4.631.5.66 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline], [inherited]`

Floating point arithmetic extractors.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

4.631.5.67 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( void *& __p ) [inline], [inherited]`

Basic arithmetic extractors.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

4.631.5.68 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.631.5.69 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::int_type basic_istream< _CharT, _Traits>::peek( void ) [inherited]`

Looking ahead in the stream.

#### Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.631.5.70 `streamsize std::ios_base::precision( ) const [inline],[inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

4.631.5.71 `streamsize std::ios_base::precision( streamsize __prec ) [inline],[inherited]`

Changing flags.

#### Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

#### Returns

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

4.631.5.72 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::put( char_type __c ) [inherited]`

Simple insertion.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**4.631.5.73** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::putback ( char_type __c ) [inherited]`

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**4.631.5.74** `void*& std::ios_base::pword ( int __ix ) [inline], [inherited]`

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

4.631.5.75 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * basic_ios<_CharT, _Traits>::rdbuf( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

4.631.5.76 `template<typename _CharT, typename _Traits, typename _Alloc> __stringbuf_type* std::basic_stringstream<_CharT, _Traits, _Alloc>::rdbuf( ) const [inline]`

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 575 of file sstream.

4.631.5.77 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline], [inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See std::ios\_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**4.631.5.78** template<typename \_CharT, typename \_Traits > **basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::read ( char\_type \* \_\_s, streamsize \_\_n )** [inherited]

Extraction without delimiters.

**Parameters**

|            |                                        |
|------------|----------------------------------------|
| <b>__s</b> | A character array.                     |
| <b>__n</b> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**4.631.5.79** template<typename \_CharT, typename \_Traits > **streamsize basic\_istream< \_CharT, \_Traits >::readsome ( char\_type \* \_\_s, streamsize \_\_n )** [inherited]

Extraction until the buffer is exhausted, but no more.

**Parameters**

|            |                                        |
|------------|----------------------------------------|
| <b>__s</b> | A character array.                     |
| <b>__n</b> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.631.5.80** void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.631.5.81** template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits > & basic\_istream< \_CharT, \_Traits >::seekg ( pos\_type \_\_pos ) [inherited]

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos (__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.631.5.82 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & basic_istream< _CharT, _Traits>::seekg( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::rdstate()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.631.5.83 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

4.631.5.84 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.631.5.85** `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline]`, `[inherited]`

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.631.5.86** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline]`, `[inherited]`

Setting new format flags.

**Parameters**

|                      |                                           |
|----------------------|-------------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                  |
| <code>__mask</code>  | The flags mask for <code>__fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `__fmtfl & mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.631.5.87** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )` `[inline]`, `[inherited]`

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**4.631.5.88** template<typename \_CharT, typename \_Traits, typename \_Alloc > \_\_string\_type std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::str ( ) const [inline]

Copying out the string buffer.

#### Returns

rdbuf ()->str ()

Definition at line 583 of file sstream.

**4.631.5.89** template<typename \_CharT, typename \_Traits, typename \_Alloc > void std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::str ( const \_\_string\_type & \_\_s ) [inline]

Setting a new buffer.

#### Parameters

|     |                                      |
|-----|--------------------------------------|
| __s | The string to use as a new sequence. |
|-----|--------------------------------------|

Calls rdbuf ()->str (s).

Definition at line 593 of file sstream.

**4.631.5.90** template<typename \_CharT, typename \_Traits > int basic\_istream< \_CharT, \_Traits >::sync ( void ) [inherited]

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If rdbuf () is a null pointer, returns -1.

Otherwise, calls rdbuf ()->pubsync (), and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount ().

Definition at line 781 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**4.631.5.91** `static bool std::ios_base::sync_with_stdio ( bool __sync = true ) [static], [inherited]`

Interaction with the standard C I/O objects.

#### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.631.5.92** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg ( void ) [inherited]`

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 817 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**4.631.5.93** `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp ( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

Definition at line 237 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

4.631.5.94 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline], [inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

4.631.5.95 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( basic_ostream<_CharT, _Traits>* __tiestr ) [inline], [inherited]`

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 301 of file `basic_ios.h`.

4.631.5.96 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & basic_istream<_CharT, _Traits>::unget ( void ) [inherited]`

Unextracting the previous character.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc()`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

4.631.5.97 void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline],[inherited]

Clearing format flags.

#### Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

4.631.5.98 template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios<\_CharT, \_Traits>::widen ( char \_\_c ) const [inline],[inherited]

Widens characters.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.-html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios<char, _Traits>::fill()`, `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::getline()`, `std::tr2::operator>>()`, and `std::operator>>()`.

4.631.5.99 streamsize std::ios\_base::width ( ) const [inline],[inherited]

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, and `std::operator>>()`.

4.631.5.100 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline],[inherited]

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

4.631.5.101 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & basic_ostream< _CharT, _Traits>::write ( const char_type * __s, streamsize __n )` `[inherited]`

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

References `std::basic_ostream< _CharT, _Traits>::_M_write()`, and `std::ios_base::badbit`.

4.631.5.102 `static int std::ios_base::xalloc ( ) throw ()` `[static], [inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 4.631.6 Member Data Documentation

**4.631.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::_M_gcount`  
`[protected], [inherited]`

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream<char>::gcount()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::basic_istream<char>::~~basic_istream()`.

**4.631.6.2** `const fmtflags std::ios_base::adjustfield` `[static], [inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.631.6.3** `const openmode std::ios_base::app` `[static], [inherited]`

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.631.6.4** `const openmode std::ios_base::ate` `[static], [inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**4.631.6.5** `const iostate std::ios_base::badbit` `[static], [inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, `std::basic_ostream<_CharT, _Traits>::write()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~~sentry()`.

**4.631.6.6** `const fmtflags std::ios_base::basefield` `[static], [inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<`

\_CharT, \_Traits >::operator<<().

#### 4.631.6.7 const seekdir std::ios\_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

#### 4.631.6.8 const openmode std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 4.631.6.9 const fmtflags std::ios\_base::boolalpha [static], [inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 258 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

#### 4.631.6.10 const seekdir std::ios\_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 4.631.6.11 const fmtflags std::ios\_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios\_base.h.

Referenced by std::dec().

#### 4.631.6.12 const seekdir std::ios\_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.631.6.13 const iostate std::ios\_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time-

`_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsomewhat()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

#### 4.631.6.14 `const ios_base::failbit` `[static]`, `[inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ifstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_fstream<_CharT, _Traits>::close()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

#### 4.631.6.15 `const fmtflags std::ios_base::fixed` `[static]`, `[inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file `ios_base.h`.

Referenced by `std::fixed()`.

#### 4.631.6.16 `const fmtflags std::ios_base::floatfield` `[static]`, `[inherited]`

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 316 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

#### 4.631.6.17 `const ios_base::goodbit` `[static]`, `[inherited]`

Indicates all is well.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsomewhat()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**4.631.6.18** `const fmtflags std::ios_base::hex` `[static], [inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**4.631.6.19** `const openmode std::ios_base::in` `[static], [inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**4.631.6.20** `const fmtflags std::ios_base::internal` `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

**4.631.6.21** `const fmtflags std::ios_base::left` `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

**4.631.6.22** `const fmtflags std::ios_base::oct` `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**4.631.6.23** `const openmode std::ios_base::out` `[static], [inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**4.631.6.24** `const fmtflags std::ios_base::right` `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

**4.631.6.25** `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

**4.631.6.26** `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.631.6.27** `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**4.631.6.28** `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**4.631.6.29** `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

**4.631.6.30** `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

**4.631.6.31** `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

**4.631.6.32** `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [sstream](#)

## 4.632 std::bernoulli\_distribution Class Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef bool [result\\_type](#)

### Public Member Functions

- [bernoulli\\_distribution](#) (double \_\_p=0.5)
- [bernoulli\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) param () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

### Friends

- bool operator== (const [bernoulli\\_distribution](#) &\_\_d1, const [bernoulli\\_distribution](#) &\_\_d2)

#### 4.632.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Definition at line 3572 of file random.h.

## 4.632.2 Member Typedef Documentation

## 4.632.2.1 typedef bool std::bernoulli\_distribution::result\_type

The type of the range of the distribution.

Definition at line 3576 of file random.h.

## 4.632.3 Constructor &amp; Destructor Documentation

## 4.632.3.1 std::bernoulli\_distribution::bernoulli\_distribution ( double \_\_p = 0.5 ) [inline], [explicit]

Constructs a Bernoulli distribution with likelihood *p*.

## Parameters

|                  |                                                                                         |
|------------------|-----------------------------------------------------------------------------------------|
| <code>__p</code> | [IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$ . |
|------------------|-----------------------------------------------------------------------------------------|

Definition at line 3609 of file random.h.

## 4.632.4 Member Function Documentation

## 4.632.4.1 result\_type std::bernoulli\_distribution::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3659 of file random.h.

References std::numeric\_limits<\_Tp>::max().

## 4.632.4.2 result\_type std::bernoulli\_distribution::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3652 of file random.h.

References std::numeric\_limits<\_Tp>::min().

## 4.632.4.3 template&lt;typename \_UniformRandomNumberGenerator&gt; result\_type std::bernoulli\_distribution::operator() ( \_UniformRandomNumberGenerator &amp; \_\_urng ) [inline]

Generating functions.

Definition at line 3667 of file random.h.

References operator()().

Referenced by operator()().

## 4.632.4.4 double std::bernoulli\_distribution::p ( ) const [inline]

Returns the *p* parameter of the distribution.

Definition at line 3630 of file random.h.

## 4.632.4.5 param\_type std::bernoulli\_distribution::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 3637 of file random.h.

Referenced by std::operator>>().

4.632.4.6 void std::bernoulli\_distribution::param ( const param\_type & \_\_param ) [inline]

Sets the parameter set of the distribution.

#### Parameters

|         |                                            |
|---------|--------------------------------------------|
| __param | The new parameter set of the distribution. |
|---------|--------------------------------------------|

Definition at line 3645 of file random.h.

4.632.4.7 void std::bernoulli\_distribution::reset ( ) [inline]

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3624 of file random.h.

#### 4.632.5 Friends And Related Function Documentation

4.632.5.1 bool operator== ( const bernoulli\_distribution & \_\_d1, const bernoulli\_distribution & \_\_d2 ) [friend]

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3709 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.633 std::bernoulli\_distribution::param\_type Struct Reference

### Public Types

- typedef [bernoulli\\_distribution](#) distribution\_type

### Public Member Functions

- **param\_type** (double \_\_p=0.5)
- double **p** () const

### Friends

- bool **operator==** (const [param\\_type](#) & \_\_p1, const [param\\_type](#) & \_\_p2)

#### 4.633.1 Detailed Description

Parameter type.

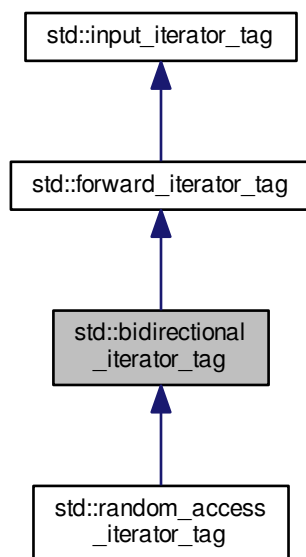
Definition at line 3578 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.634 std::bidirectional\_iterator\_tag Struct Reference

Inheritance diagram for std::bidirectional\_iterator\_tag:



##### 4.634.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.

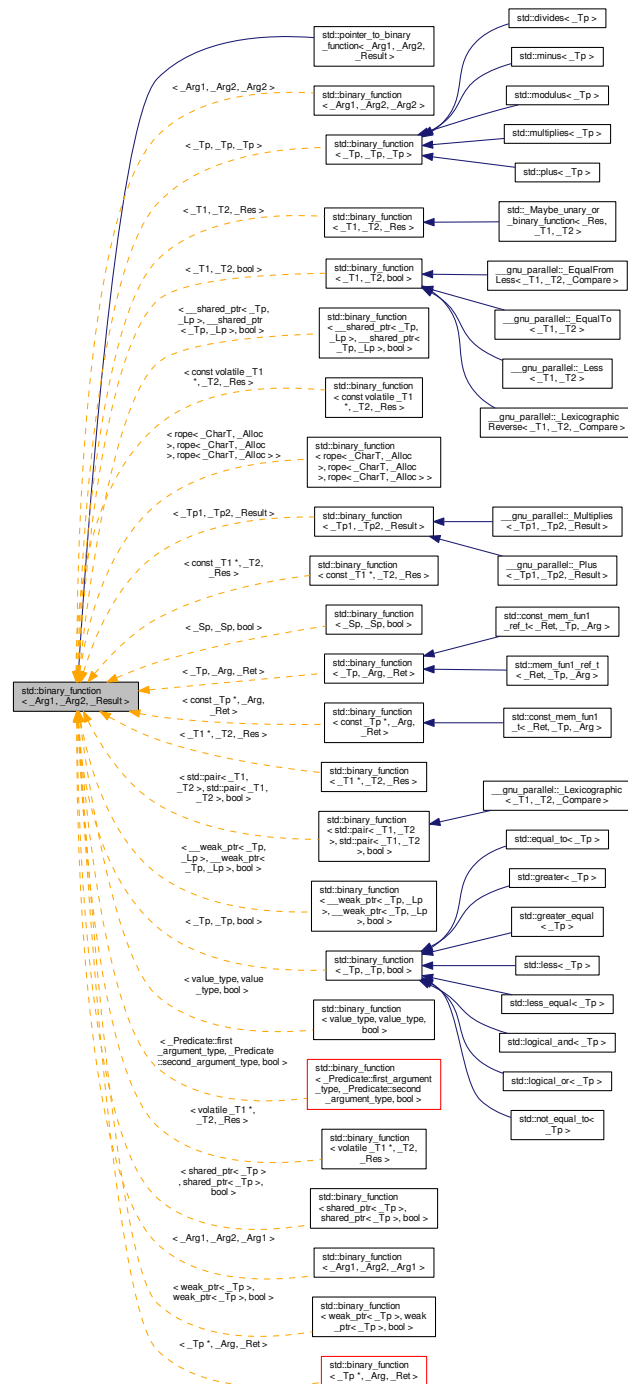
Definition at line 99 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.635 std::binary\_function&lt; \_Arg1, \_Arg2, \_Result &gt; Struct Template Reference

Inheritance diagram for std::binary\_function< \_Arg1, \_Arg2, \_Result >:



## Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

## 4.635.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> struct std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 114 of file stl\_function.h.

## 4.635.2 Member Typedef Documentation

4.635.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.635.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type`

`result_type` is the return type

Definition at line 123 of file stl\_function.h.

4.635.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type`

`second_argument_type` is the type of the second argument

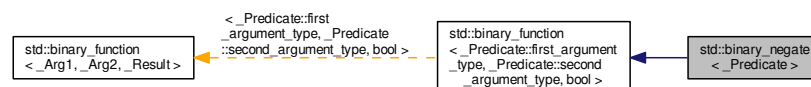
Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.636 std::binary\_negate&lt; \_Predicate &gt; Class Template Reference

Inheritance diagram for `std::binary_negate< _Predicate >`:



## Public Types

- typedef  
    \_Predicate::first\_argument\_type [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef  
    \_Predicate::second\_argument\_type [second\\_argument\\_type](#)

## Public Member Functions

- **binary\_negate** (const \_Predicate &\_\_x)
- bool **operator()** (const typename \_Predicate::first\_argument\_type &\_\_x, const typename \_Predicate::second\_argument\_type &\_\_y) const

## Protected Attributes

- \_Predicate **\_M\_pred**

### 4.636.1 Detailed Description

`template<typename _Predicate>class std::binary_negate<_Predicate>`

One of the [negation functors](#).

Definition at line 374 of file `stl_function.h`.

### 4.636.2 Member Typedef Documentation

4.636.2.1 `typedef _Predicate::first_argument_type std::binary_function<_Predicate::first_argument_type ,  
_Predicate::second_argument_type , bool >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.636.2.2 `typedef bool std::binary_function<_Predicate::first_argument_type ,_Predicate::second_argument_type , bool  
>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.636.2.3 `typedef _Predicate::second_argument_type std::binary_function<_Predicate::first_argument_type ,  
_Predicate::second_argument_type , bool >::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

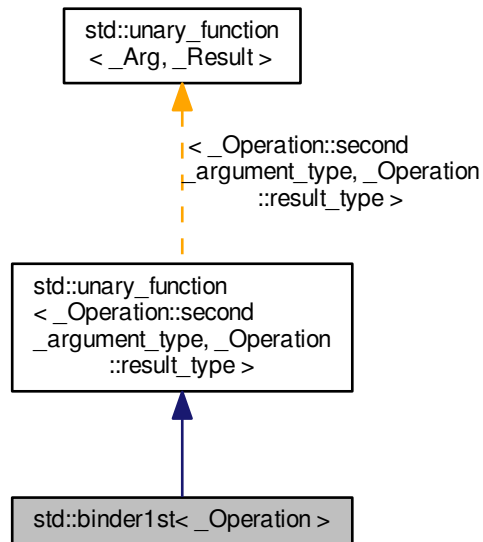
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.637 std::binder1st&lt; \_Operation &gt; Class Template Reference

Inheritance diagram for std::binder1st< \_Operation >:



## Public Types

- typedef `_Operation::second_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

## Public Member Functions

- **binder1st** (const `_Operation` &\_\_x, const typename `_Operation::first_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &\_\_x) const

## Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

## 4.637.1 Detailed Description

```
template<typename _Operation>class std::binder1st< _Operation >
```

One of the [binder functors](#).

Definition at line 104 of file binders.h.

#### 4.637.2 Member Typedef Documentation

4.637.2.1 `typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type ,  
_Operation::result_type >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl\_function.h.

4.637.2.2 `typedef _Operation::result_type std::unary_function< _Operation::second_argument_type , _Operation::result_type  
>::result_type` [inherited]

`result_type` is the return type

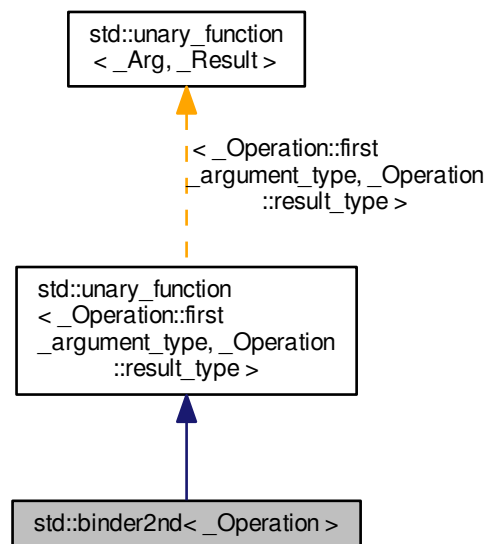
Definition at line 107 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [binders.h](#)

#### 4.638 std::binder2nd< \_Operation > Class Template Reference

Inheritance diagram for `std::binder2nd< _Operation >`:



## Public Types

- typedef `_Operation::first_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

## Public Member Functions

- **binder2nd** (const `_Operation` &\_\_x, const typename `_Operation::second_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::first_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::first_argument_type` &\_\_x) const

## Protected Attributes

- `_Operation` **op**
- `_Operation::second_argument_type` **value**

## 4.638.1 Detailed Description

`template<typename _Operation>class std::binder2nd< _Operation >`

One of the [binder functors](#).

Definition at line 139 of file `binders.h`.

## 4.638.2 Member Typedef Documentation

4.638.2.1 `typedef _Operation::first_argument_type std::unary_function< _Operation::first_argument_type ,  
_Operation::result_type >::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.638.2.2 `typedef _Operation::result_type std::unary_function< _Operation::first_argument_type , _Operation::result_type  
>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

4.639 `std::binomial_distribution< _IntType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- **binomial\_distribution** (\_IntType \_\_t=\_IntType(1), double \_\_p=0.5)
- **binomial\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- \_IntType **t** () const

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::binomial\\_distribution](#)< \_IntType1 > &\_\_x)
- bool **operator==** (const [binomial\\_distribution](#) &\_\_d1, const [binomial\\_distribution](#) &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::binomial\\_distribution](#)< \_IntType1 > &\_\_x)

## 4.639.1 Detailed Description

template<typename \_IntType = int>class [std::binomial\\_distribution](#)< \_IntType >

A discrete binomial random number distribution.

The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3777 of file random.h.

## 4.639.2 Member Typedef Documentation

4.639.2.1 `template<typename _IntType = int> typedef _IntType std::binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3780 of file random.h.

## 4.639.3 Member Function Documentation

4.639.3.1 `template<typename _IntType = int> result_type std::binomial_distribution< _IntType >::max ( ) const`  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 3887 of file random.h.

4.639.3.2 `template<typename _IntType = int> result_type std::binomial_distribution< _IntType >::min ( ) const`  
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3880 of file random.h.

4.639.3.3 `template<typename _IntType = int> template<typename UniformRandomNumberGenerator > result_type`  
`std::binomial_distribution< _IntType >::operator() ( UniformRandomNumberGenerator & __urng )` [inline]

Generating functions.

Definition at line 3895 of file random.h.

References `std::binomial_distribution< _IntType >::operator()()`.

Referenced by `std::binomial_distribution< _IntType >::operator()()`.

4.639.3.4 `template<typename _IntType > template<typename UniformRandomNumberGenerator > binomial_distribution<`  
`_IntType >::result_type std::binomial_distribution< _IntType >::operator() ( UniformRandomNumberGenerator`  
`& __urng, const param_type & __param )`

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise.  
NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1685 of file bits/random.tcc.

References `std::abs()`, `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::numeric_limits< _Tp >::max()`.

4.639.3.5 `template<typename _IntType = int> double std::binomial_distribution< _IntType >::p ( ) const` [inline]

Returns the distribution `p` parameter.

Definition at line 3858 of file random.h.

4.639.3.6 `template<typename _IntType = int> param_type std::binomial_distribution< _IntType >::param ( ) const`  
[inline]

Returns the parameter set of the distribution.

Definition at line 3865 of file random.h.

4.639.3.7 `template<typename _IntType = int> void std::binomial_distribution< _IntType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3873 of file random.h.

4.639.3.8 `template<typename _IntType = int> void std::binomial_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3844 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

4.639.3.9 `template<typename _IntType = int> _IntType std::binomial_distribution< _IntType >::t ( ) const [inline]`

Returns the distribution `t` parameter.

Definition at line 3851 of file random.h.

#### 4.639.4 Friends And Related Function Documentation

4.639.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_ostream< _CharT , _Traits> & operator<< ( std::basic_ostream< _CharT , _Traits > & __os, const std::binomial_distribution< _IntType1 > & __x ) [friend]`

Inserts a `binomial_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>binomial_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.639.4.2 `template<typename _IntType = int> bool operator== ( const binomial_distribution< _IntType > & __d1, const binomial_distribution< _IntType > & __d2 ) [friend]`

Return true if two `binomial` distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3931 of file random.h.

4.639.4.3 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_istream< _CharT , _Traits> & operator>> ( std::basic_istream< _CharT , _Traits > & __is, std::binomial_distribution< _IntType1 > & __x ) [friend]`

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                     |
| <code>__x</code>  | A <code>binomial_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.640 `std::binomial_distribution< _IntType >::param_type` Struct Reference

## Public Types

- typedef [binomial\\_distribution](#)  
< `_IntType` > **distribution\_type**

## Public Member Functions

- **param\_type** (`_IntType` `__t`=`_IntType`(1), double `__p`=0.5)
- double **p** () const
- `_IntType` **t** () const

## Friends

- class **binomial\_distribution**< `_IntType` >
- bool **operator==** (const [param\\_type](#) &`__p1`, const [param\\_type](#) &`__p2`)

## 4.640.1 Detailed Description

```
template<typename _IntType = int>struct std::binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3786 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.641 `std::cauchy_distribution< _RealType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **cauchy\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **cauchy\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [cauchy\\_distribution](#) &\_\_d1, const [cauchy\\_distribution](#) &\_\_d2)

## 4.641.1 Detailed Description

template<typename \_RealType = double>class std::cauchy\_distribution<\_RealType>

A cauchy\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|a,b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2929 of file random.h.

## 4.641.2 Member Typedef Documentation

4.641.2.1 template<typename \_RealType = double> typedef \_RealType std::cauchy\_distribution<\_RealType>::result\_type

The type of the range of the distribution.

Definition at line 2932 of file random.h.

## 4.641.3 Member Function Documentation

4.641.3.1 `template<typename _RealType = double> result_type std::cauchy_distribution<_RealType>::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3020 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

4.641.3.2 `template<typename _RealType = double> result_type std::cauchy_distribution<_RealType>::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3013 of file random.h.

References `std::numeric_limits<_Tp>::min()`.

4.641.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type`  
`std::cauchy_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3028 of file random.h.

References `std::cauchy_distribution<_RealType>::operator()()`.

Referenced by `std::cauchy_distribution<_RealType>::operator()()`.

4.641.3.4 `template<typename _RealType = double> param_type std::cauchy_distribution<_RealType>::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 2998 of file random.h.

Referenced by `std::operator>>()`.

4.641.3.5 `template<typename _RealType = double> void std::cauchy_distribution<_RealType>::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3006 of file random.h.

4.641.3.6 `template<typename _RealType = double> void std::cauchy_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2980 of file random.h.

## 4.641.4 Friends And Related Function Documentation

4.641.4.1 `template<typename _RealType = double> bool operator==( const cauchy_distribution<_RealType> & __d1, const cauchy_distribution<_RealType> & __d2 ) [friend]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 3063 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.642 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [cauchy\\_distribution](#)  
<\_RealType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 4.642.1 Detailed Description

`template<typename _RealType = double> struct std::cauchy_distribution<_RealType>::param_type`

Parameter type.

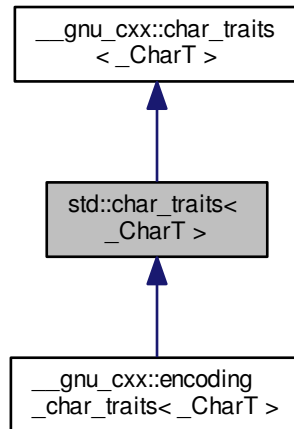
Definition at line 2938 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.643 std::char\_traits&lt;\_CharT&gt; Struct Template Reference

Inheritance diagram for std::char\_traits<\_CharT>:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

## Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **eof** ()
- static constexpr bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static constexpr bool **eq\_int\_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)
- static `std::size_t` **length** (`const char_type *__s`)
- static constexpr bool **lt** (`const char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **not\_eof** (`const int_type &__c`)

- static constexpr `char_type` **to\_char\_type** (const `int_type` &\_\_c)
- static constexpr `int_type` **to\_int\_type** (const `char_type` &\_\_c)

#### 4.643.1 Detailed Description

`template<class _CharT> struct std::char_traits< _CharT >`

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 227 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.644 `std::char_traits< __gnu_cxx::character< V, I, S > >` Struct Template Reference

### Public Types

- typedef `__gnu_cxx::character< V, I, S >` **char\_type**
- typedef `char_type::int_type` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `fpos< state_type >` **pos\_type**
- typedef `char_type::state_type` **state\_type**

### Static Public Member Functions

- static void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **assign** (`char_type` \* \_\_s, `size_t` \_\_n, `char_type` \_\_a)
- static `int` **compare** (const `char_type` \* \_\_s1, const `char_type` \* \_\_s2, `size_t` \_\_n)
- static `char_type` \* **copy** (`char_type` \* \_\_s1, const `char_type` \* \_\_s2, `size_t` \_\_n)
- static `int_type` **eof** ()
- static `bool` **eq** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `bool` **eq\_int\_type** (const `int_type` &\_\_c1, const `int_type` &\_\_c2)
- static const `char_type` \* **find** (const `char_type` \* \_\_s, `size_t` \_\_n, const `char_type` &\_\_a)
- static `size_t` **length** (const `char_type` \* \_\_s)
- static `bool` **lt** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **move** (`char_type` \* \_\_s1, const `char_type` \* \_\_s2, `size_t` \_\_n)
- static `int_type` **not\_eof** (const `int_type` &\_\_c)
- static `char_type` **to\_char\_type** (const `int_type` &\_\_i)
- static `int_type` **to\_int\_type** (const `char_type` &\_\_c)

## 4.644.1 Detailed Description

```
template<typename V, typename I, typename S>struct std::char_traits< __gnu_cxx::character< V, I, S > >
```

char\_traits<\_\_gnu\_cxx::character> specialization.

Definition at line 95 of file pod\_char\_traits.h.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 4.645 std::char\_traits&lt; char &gt; Struct Template Reference

## Public Types

- typedef char **char\_type**
- typedef int **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [streampos](#) **pos\_type**
- typedef mbstate\_t **state\_type**

## Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **assign** (char\_type \* \_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static const char\_type \* **find** (const char\_type \* \_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

## 4.645.1 Detailed Description

```
template<>struct std::char_traits< char >
```

## 21.1.3.1 char\_traits specializations

Definition at line 233 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.646 std::char\_traits&lt; wchar\_t &gt; Struct Template Reference

## Public Types

- typedef wchar\_t **char\_type**
- typedef wint\_t **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [wstreampos](#) **pos\_type**
- typedef mbstate\_t **state\_type**

## Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

## 4.646.1 Detailed Description

```
template<>struct std::char_traits< wchar_t >
```

## 21.1.3.2 char\_traits specializations

Definition at line 304 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.647 std::chi\_squared\_distribution&lt; \_RealType &gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **chi\_squared\_distribution** (\_RealType \_\_n=\_RealType(1))
- **chi\_squared\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- \_RealType **n** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [chi\\_squared\\_distribution](#) &\_\_d1, const [chi\\_squared\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)

## 4.647.1 Detailed Description

template<typename \_RealType = double>class std::chi\_squared\_distribution<\_RealType>

A chi\_squared\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1}e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

Definition at line 2719 of file random.h.

## 4.647.2 Member Typedef Documentation

4.647.2.1 `template<typename _RealType = double> typedef _RealType std::chi_squared_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2722 of file random.h.

#### 4.647.3 Member Function Documentation

4.647.3.1 `template<typename _RealType = double> result_type std::chi_squared_distribution< _RealType >::max ( )  
const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2799 of file random.h.

References std::numeric\_limits<\_Tp>::max().

4.647.3.2 `template<typename _RealType = double> result_type std::chi_squared_distribution< _RealType >::min ( )  
const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2792 of file random.h.

4.647.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type  
std::chi_squared_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng )  
[inline]`

Generating functions.

Definition at line 2807 of file random.h.

4.647.3.4 `template<typename _RealType = double> param_type std::chi_squared_distribution< _RealType >::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 2777 of file random.h.

4.647.3.5 `template<typename _RealType = double> void std::chi_squared_distribution< _RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2785 of file random.h.

4.647.3.6 `template<typename _RealType = double> void std::chi_squared_distribution< _RealType >::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 2763 of file random.h.

References std::gamma\_distribution<\_RealType>::reset().

## 4.647.4 Friends And Related Function Documentation

4.647.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( const std::basic_ostream<_CharT, _Traits> & __os, const std::chi_squared_distribution<_RealType1> & __x ) [friend]`

Inserts a chi\_squared\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A chi_squared_distribution random number distribution. |

## Returns

The output stream with the state of \_\_x inserted or in an error state.

4.647.4.2 `template<typename _RealType = double> bool operator== ( const chi_squared_distribution<_RealType> & __d1, const chi_squared_distribution<_RealType> & __d2 ) [friend]`

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2858 of file random.h.

4.647.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::chi_squared_distribution<_RealType1> & __x ) [friend]`

Extracts a chi\_squared\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__is</code> | An input stream.                                           |
| <code>__x</code>  | A chi_squared_distribution random number generator engine. |

## Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.648 std::chi\_squared\_distribution&lt;\_RealType&gt;::param\_type Struct Reference

## Public Types

- typedef  
[chi\\_squared\\_distribution](#)  
<\_RealType> **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_n=\_RealType(1))
- \_RealType **n** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.648.1 Detailed Description

template<typename \_RealType = double>struct std::chi\_squared\_distribution< \_RealType >::param\_type

Parameter type.

Definition at line 2728 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.649 std::chrono::duration&lt; \_Rep, \_Period &gt; Struct Template Reference

## Public Types

- typedef \_Period **period**
- typedef \_Rep **rep**

## Public Member Functions

- **duration** (const [duration](#) &)=default
- template<typename \_Rep2, typename = typename enable\_if<is\_convertible<\_Rep2, rep>::value && (treat\_as\_floating\_point<rep>::value || !treat\_as\_floating\_point<\_Rep2>::value)>::type>  
constexpr **duration** (const \_Rep2 &\_\_rep)
- template<typename \_Rep2, typename \_Period2, typename = typename enable\_if<treat\_as\_floating\_point<rep>::value || (ratio\_divide<\_Period2, period>::den == 1 && !treat\_as\_floating\_point<\_Rep2>::value)>::type>  
constexpr **duration** (const [duration](#)< \_Rep2, \_Period2 > &\_\_d)
- constexpr rep **count** () const
- template<typename \_Rep2 = rep>  
[enable\\_if](#)  
< !treat\_as\_floating\_point  
< \_Rep2 >::value, [duration](#) & >  
::type **operator%=>** (const rep &\_\_rhs)
- template<typename \_Rep2 = rep>  
[enable\\_if](#)  
< !treat\_as\_floating\_point  
< \_Rep2 >::value, [duration](#) & >  
::type **operator%=>** (const [duration](#) &\_\_d)
- [duration](#) & **operator\*=>** (const rep &\_\_rhs)
- constexpr [duration](#) **operator+>** () const
- [duration](#) & **operator++>** ()

- `duration operator++` (int)
- `duration & operator+=` (const `duration` &\_\_d)
- constexpr `duration operator-` () const
- `duration & operator--` ()
- `duration operator--` (int)
- `duration & operator-=` (const `duration` &\_\_d)
- `duration & operator/=` (const rep &\_\_rhs)
- `duration & operator=` (const `duration` &)=default

#### Static Public Member Functions

- static constexpr `duration max` ()
- static constexpr `duration min` ()
- static constexpr `duration zero` ()

##### 4.649.1 Detailed Description

`template<typename _Rep, typename _Period> struct std::chrono::duration< _Rep, _Period >`

`duration`

Definition at line 240 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

#### 4.650 `std::chrono::duration_values<_Rep>` Struct Template Reference

##### Static Public Member Functions

- static constexpr `_Rep max` ()
- static constexpr `_Rep min` ()
- static constexpr `_Rep zero` ()

##### 4.650.1 Detailed Description

`template<typename _Rep> struct std::chrono::duration_values< _Rep >`

`duration_values`

Definition at line 213 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

## 4.651 std::chrono::system\_clock Struct Reference

### Public Types

- typedef [chrono::seconds](#) **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef [chrono::time\\_point](#)  
< [system\\_clock](#), [duration](#) > **time\_point**

### Static Public Member Functions

- static [time\\_point](#) **from\_time\_t** (std::time\_t \_\_t) noexcept
- static [time\\_point](#) **now** () noexcept
- static std::time\_t **to\_time\_t** (const [time\\_point](#) &\_\_t) noexcept

### Static Public Attributes

- static constexpr bool **is\_steady**

#### 4.651.1 Detailed Description

system\_clock

Definition at line 690 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 4.652 std::chrono::time\_point< \_Clock, \_Dur > Struct Template Reference

### Public Types

- typedef \_Clock **clock**
- typedef \_Dur **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**

### Public Member Functions

- constexpr [time\\_point](#) (const duration &\_\_dur)
- template<typename \_Dur2 >  
constexpr [time\\_point](#) (const [time\\_point](#)< clock, \_Dur2 > &\_\_t)
- [time\\_point](#) & **operator+=** (const duration &\_\_dur)
- [time\\_point](#) & **operator-=** (const duration &\_\_dur)
- constexpr duration **time\_since\_epoch** () const

## Static Public Member Functions

- static constexpr [time\\_point](#) **max** ()
- static constexpr [time\\_point](#) **min** ()

## 4.652.1 Detailed Description

```
template<typename _Clock, typename _Dur>struct std::chrono::time_point< _Clock, _Dur >
```

[time\\_point](#)

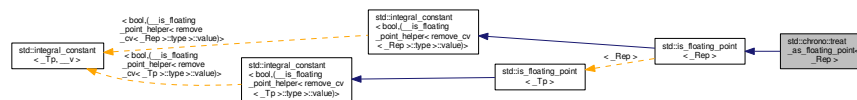
Definition at line 545 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 4.653 std::chrono::treat\_as\_floating\_point&lt; \_Rep &gt; Struct Template Reference

Inheritance diagram for std::chrono::treat\_as\_floating\_point< \_Rep >:



## Public Types

- typedef [integral\\_constant](#)  
< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.653.1 Detailed Description

```
template<typename _Rep>struct std::chrono::treat_as_floating_point< _Rep >
```

[treat\\_as\\_floating\\_point](#)

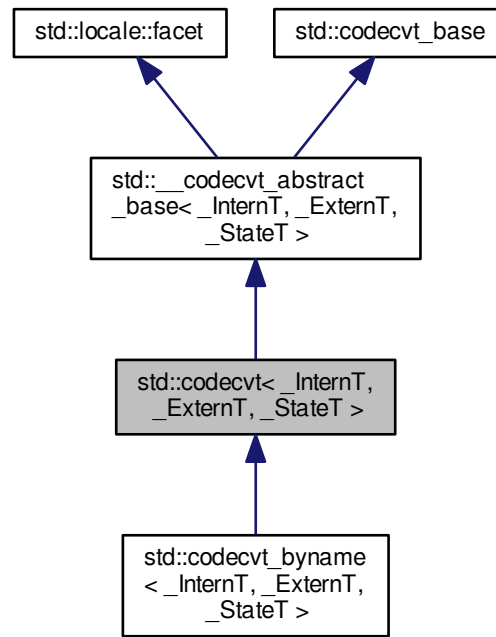
Definition at line 207 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 4.654 std::codecvt&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::codecvt< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \* \_\_&\_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* \_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* \_\_&\_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.654.1 Detailed Description

template<typename \_InternT, typename \_ExternT, typename \_StateT>class std::codecvt< \_InternT, \_ExternT, \_StateT >

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 276 of file codecvt.h.

## 4.654.2 Member Function Documentation

4.654.2.1 template<typename \_InternT, typename \_ExternT, typename \_StateT > virtual result std::codecvt< \_InternT, \_ExternT, \_StateT >::do\_out ( state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* \_\_&\_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_&\_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See Also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

4.654.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const [inline], [inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

4.654.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [inline], [inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

4.654.2.4 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` `[inline]`, `[inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

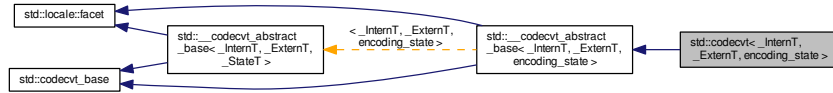
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.655 std::codecvt&lt; \_InternT, \_ExternT, encoding\_state &gt; Class Template Reference

Inheritance diagram for std::codecvt< \_InternT, \_ExternT, encoding\_state >:



## Public Types

- typedef `state_type::descriptor_type` **descriptor\_type**
- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state\_type**

## Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`state_type &__enc, size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__from_next, intern_type *__to, intern_type *__to_end, intern_type *__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- **int max\_length** () const throw ()
- **result out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__from_next, extern_type *__to, extern_type *__to_end, extern_type *__to_next`) const
- **result unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__to_next`) const

## Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual **bool do\_always\_noconv** () const throw ()
- virtual **int do\_encoding** () const throw ()
- virtual **result do\_in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__from_next, intern_type *__to, intern_type *__to_end, intern_type *__to_next`) const
- virtual **int do\_length** (`state_type &, const extern_type *__from, const extern_type *__end, size_t __max`) const
- virtual **int do\_max\_length** () const throw ()
- virtual **result do\_out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__from_next, extern_type *__to, extern_type *__to_end, extern_type *__to_next`) const
- virtual **result do\_unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__to_next`) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.655.1 Detailed Description

template<typename \_InternT, typename \_ExternT>class std::codecvt< \_InternT, \_ExternT, encoding\_state >

codecvt<InternT, \_ExternT, encoding\_state> specialization.

Definition at line 230 of file codecvt\_specializations.h.

## 4.655.2 Member Function Documentation

4.655.2.1 template<typename \_InternT, typename \_ExternT > codecvt\_base::result std::codecvt< \_InternT, \_ExternT, encoding\_state >::do\_out ( state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* & \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* & \_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See Also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

Definition at line 306 of file codecvt\_specializations.h.

4.655.2.2 result std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, encoding\_state >::in ( state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* & \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* & \_\_to\_next ) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

**4.655.2.3** `result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const` `[inline], [inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

References std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >::do\_out().

4.655.2.4 result std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, encoding\_state >::unshift ( state\_type & \_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to\_next ) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

## Parameters

|           |                                      |
|-----------|--------------------------------------|
| __state   | Persistent conversion state data.    |
| __to      | Start of output buffer.              |
| __to_end  | End of output buffer.                |
| __to_next | Returns start of unused output area. |

## Returns

codecvt\_base::result.

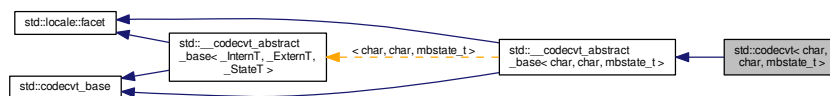
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 4.656 std::codecvt&lt; char, char, mbstate\_t &gt; Class Template Reference

Inheritance diagram for std::codecvt< char, char, mbstate\_t >:



## Public Types

- typedef char **extern\_type**

- typedef char **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

#### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

#### Static Public Attributes

- static [locale::id](#) **id**

#### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

#### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_ic\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.656.1 Detailed Description

template<>class std::codecvt< char, char, mbstate\_t >

class codecvt<char, char, mbstate\_t> specialization.

Definition at line 340 of file codecvt.h.

## 4.656.2 Member Function Documentation

4.656.2.1 virtual result std::codecvt< char, char, mbstate\_t >::do\_out ( state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*& \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See Also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

4.656.2.2 result std::\_\_codecvt\_abstract\_base< char, char, mbstate\_t >::in ( state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*& \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*& \_\_to\_next ) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

## Parameters

|             |                                      |
|-------------|--------------------------------------|
| __state     | Persistent conversion state data.    |
| __from      | Start of input.                      |
| __from_end  | End of input.                        |
| __from_next | Returns start of unconverted data.   |
| __to        | Start of output buffer.              |
| __to_end    | End of output buffer.                |
| __to_next   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

Definition at line 196 of file codecvt.h.

**4.656.2.3** `result std::__codecvt_abstract_base< char, char, mbstate_t >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` `[inline], [inherited]`

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**4.656.2.4** `result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` `[inline], [inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

## Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

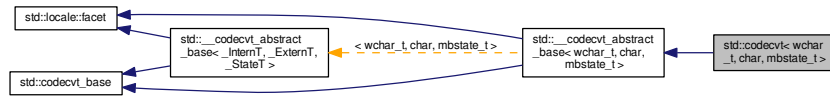
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.657 std::codecvt&lt; wchar\_t, char, mbstate\_t &gt; Class Template Reference

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



## Public Types

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef mbstate\_t **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static [locale::id](#) **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.657.1 Detailed Description

template<>class std::codecvt< wchar\_t, char, mbstate\_t >

class codecvt<wchar\_t, char, mbstate\_t> specialization.

Definition at line 398 of file codecvt.h.

## 4.657.2 Member Function Documentation

- 4.657.2.1 virtual result std::codecvt< wchar\_t, char, mbstate\_t >::do\_out ( state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*& \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to\_next ) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See Also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**4.657.2.2** `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const` `[inline], [inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

**4.657.2.3** `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const` `[inline], [inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

**4.657.2.4** `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

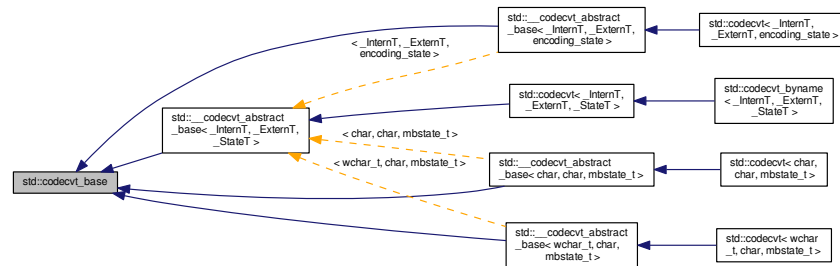
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.658 std::codecvt\_base Class Reference

Inheritance diagram for std::codecvt\_base:



## Public Types

- enum **result** { **ok**, **partial**, **error**, **noconv** }

## 4.658.1 Detailed Description

Empty base class for codecvt facet [22.2.1.5].

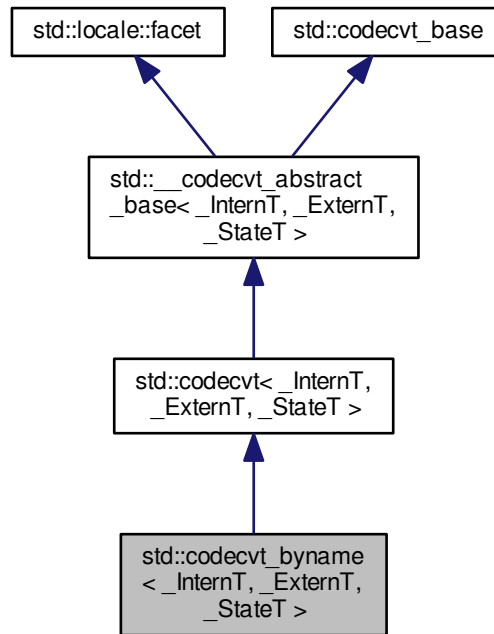
Definition at line 46 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.659 std::codecvt\_byname&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- **codecvt\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static [locale::id](#) **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.659.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class codecvt\_byname [22.2.1.6].

Definition at line 458 of file codecvt.h.

## 4.659.2 Member Function Documentation

4.659.2.1 `template<typename _InternT, typename _ExternT, typename _StateT > virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [protected], [virtual], [inherited]

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

## See Also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

4.659.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const [inline], [inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

4.659.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [inline], [inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

4.659.2.4 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` `[inline]`, `[inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

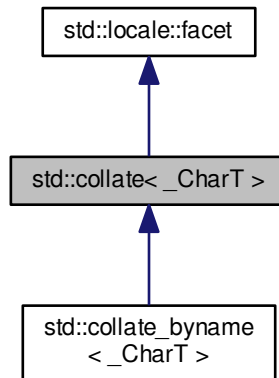
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.660 `std::collate<_CharT>` Class Template Reference

Inheritance diagram for `std::collate<_CharT>`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

## Public Member Functions

- [collate](#) (size\_t \_\_refs=0)
- [collate](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<>  
int **\_M\_compare** (const char \*, const char \*) const throw()
- template<>  
int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- template<>  
size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<>  
size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- int [compare](#) (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const
- long [hash](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const
- [string\\_type transform](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const

## Static Public Attributes

- static [locale::id](#) [id](#)

## Protected Member Functions

- virtual `~collate()`
- virtual `int do_compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const`
- virtual `long do_hash(const _CharT * __lo, const _CharT * __hi) const`
- virtual `string_type do_transform(const _CharT * __lo, const _CharT * __hi) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale(__c_locale & __cloc) throw()`
- static `void _S_create_c_locale(__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static `void _S_destroy_c_locale(__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale()`
- static `const char * _S_get_c_name() throw()`
- static `__c_locale _S_lc_type_c_locale(__c_locale __cloc, const char * __s)`

## Protected Attributes

- `__c_locale _M_c_locale_collate`

## 4.660.1 Detailed Description

`template<typename _CharT> class std::collate<_CharT>`

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 584 of file `locale_classes.h`.

## 4.660.2 Member Typedef Documentation

4.660.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Definition at line 590 of file `locale_classes.h`.

4.660.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Definition at line 591 of file `locale_classes.h`.

## 4.660.3 Constructor &amp; Destructor Documentation

4.660.3.1 `template<typename _CharT> std::collate<_CharT>::collate(size_t __refs = 0) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 611 of file locale\_classes.h.

4.660.3.2 `template<typename _CharT> std::collate<_CharT>::collate ( _c.locale __cloc, size_t __refs = 0 ) [inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__refs</code> | Passed to the base facet class. |

Definition at line 625 of file locale\_classes.h.

4.660.3.3 `template<typename _CharT> virtual std::collate<_CharT>::~~collate ( ) [inline], [protected], [virtual]`

Destructor.

Definition at line 688 of file locale\_classes.h.

#### 4.660.4 Member Function Documentation

4.660.4.1 `template<typename _CharT> int std::collate<_CharT>::compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const [inline]`

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

#### Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

#### Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 642 of file locale\_classes.h.

4.660.4.2 `template<typename _CharT> int std::collate<_CharT>::do_compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const [protected], [virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See Also**

compare().

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 161 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

**4.660.4.3** `template<typename _CharT> long std::collate<_CharT>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const`  
`[protected], [virtual]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

**Returns**

Hash value.

Definition at line 256 of file locale\_classes.tcc.

**4.660.4.4** `template<typename _CharT> collate<_CharT>::string_type std::collate<_CharT>::do_transform ( const`  
`_CharT * __lo, const _CharT * __hi ) const` `[protected], [virtual]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

|                   |        |
|-------------------|--------|
| <code>__lo</code> | Start. |
| <code>__hi</code> | End.   |

**Returns**

transformed string.

Definition at line 200 of file locale\_classes.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::append(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::data(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::length(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::push\_back().

**4.660.4.5** template<typename \_CharT> long std::collate< \_CharT >::hash ( const \_CharT \* \_\_lo, const \_CharT \* \_\_hi ) const [inline]

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning collate::do\_hash().

#### Parameters

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

#### Returns

Hash value.

Definition at line 675 of file locale\_classes.h.

**4.660.4.6** template<typename \_CharT> string\_type std::collate< \_CharT >::transform ( const \_CharT \* \_\_lo, const \_CharT \* \_\_hi ) const [inline]

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning collate::do\_transform().

#### Parameters

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

#### Returns

Transformed string\_type.

Definition at line 661 of file locale\_classes.h.

### 4.660.5 Member Data Documentation

**4.660.5.1** template<typename \_CharT> locale::id std::collate< \_CharT >::id [static]

Numpunct facet id.

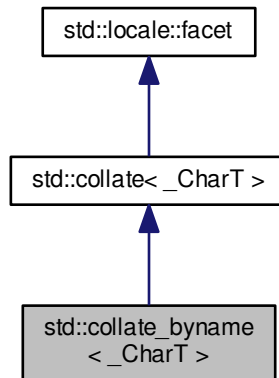
Definition at line 601 of file locale\_classes.h.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 4.661 std::collate\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::collate\_byname<\_CharT>:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string](#)<`_CharT`> [string\\_type](#)

## Public Member Functions

- **collate\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<>  
int **\_M\_compare** (const char \*, const char \*) const throw()
- template<>  
int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- template<>  
size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<>  
size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- int [compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- long [hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- [string\\_type transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

## Static Public Attributes

- static [locale::id](#) [id](#)

## Protected Member Functions

- virtual int [do\\_compare](#) (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_type\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_collate](#)

## 4.661.1 Detailed Description

template<typename \_CharT> class std::collate\_byname<\_CharT>

class collate\_byname [22.2.4.2].

Definition at line 758 of file locale\_classes.h.

## 4.661.2 Member Typedef Documentation

4.661.2.1 template<typename \_CharT> typedef \_CharT std::collate\_byname<\_CharT>::char\_type

Public typedefs.

Definition at line 763 of file locale\_classes.h.

4.661.2.2 template<typename \_CharT> typedef basic\_string<\_CharT> std::collate\_byname<\_CharT>::string\_type

Public typedefs.

Definition at line 764 of file locale\_classes.h.

## 4.661.3 Member Function Documentation

4.661.3.1 template<typename \_CharT> int std::collate<\_CharT>::compare ( const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2 ) const [inline], [inherited]

Compare two strings.

This function compares two strings and returns the result by calling collate::do\_compare().

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 642 of file locale\_classes.h.

4.661.3.2 `template<typename _CharT> int std::collate<_CharT>::do_compare ( const _CharT* __lo1, const _CharT* __hi1, const _CharT* __lo2, const _CharT* __hi2 ) const` [protected], [virtual], [inherited]

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See Also**

`compare()`.

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 161 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

4.661.3.3 `template<typename _CharT> long std::collate<_CharT>::do_hash ( const _CharT* __lo, const _CharT* __hi ) const` [protected], [virtual], [inherited]

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

**Returns**

Hash value.

Definition at line 256 of file locale\_classes.tcc.

**4.661.3.4** `template<typename _CharT> collate< _CharT >::string_type std::collate< _CharT >::do_transform ( const _CharT * __lo, const _CharT * __hi ) const` `[protected]`, `[virtual]`, `[inherited]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

|                   |        |
|-------------------|--------|
| <code>__lo</code> | Start. |
| <code>__hi</code> | End.   |

**Returns**

transformed string.

Definition at line 200 of file locale\_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, and `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`.

**4.661.3.5** `template<typename _CharT> long std::collate< _CharT >::hash ( const _CharT * __lo, const _CharT * __hi ) const` `[inline]`, `[inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

**Parameters**

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

**Returns**

Hash value.

Definition at line 675 of file locale\_classes.h.

**4.661.3.6** `template<typename _CharT> string_type std::collate< _CharT >::transform ( const _CharT * __lo, const _CharT * __hi ) const` `[inline]`, `[inherited]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

## Parameters

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

## Returns

Transformed string\_type.

Definition at line 661 of file locale\_classes.h.

## 4.661.4 Member Data Documentation

4.661.4.1 `template<typename _CharT> locale::id std::collate< _CharT >::id` `[static], [inherited]`

Numpunct facet id.

Definition at line 601 of file locale\_classes.h.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 4.662 std::complex&lt; \_Tp &gt; Struct Template Reference

## Public Types

- `typedef _Tp value_type`

## Public Member Functions

- `constexpr complex (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())`
- `template<typename _Up >`  
`constexpr complex (const complex< _Up > &__z)`
- `__attribute__((abi_tag__("cxx11"))) const expr _Tp real()`
- `__attribute__((abi_tag__("cxx11"))) const expr _Tp imag()`
- `constexpr complex __rep () const`
- `void imag (_Tp __val)`
- `complex< _Tp > & operator*= (const _Tp &)`
- `template<typename _Up >`  
`complex< _Tp > & operator*= (const complex< _Up > &)`
- `complex< _Tp > & operator+= (const _Tp &__t)`
- `template<typename _Up >`  
`complex< _Tp > & operator+= (const complex< _Up > &)`
- `complex< _Tp > & operator-= (const _Tp &__t)`
- `template<typename _Up >`  
`complex< _Tp > & operator-= (const complex< _Up > &)`
- `complex< _Tp > & operator/= (const _Tp &)`
- `template<typename _Up >`  
`complex< _Tp > & operator/= (const complex< _Up > &)`
- `complex< _Tp > & operator= (const _Tp &)`
- `template<typename _Up >`  
`complex< _Tp > & operator= (const complex< _Up > &)`
- `void real (_Tp __val)`

## 4.662.1 Detailed Description

```
template<typename _Tp> struct std::complex< _Tp >
```

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

## Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>Tp</i> | Type of real and imaginary values. |
|-----------|------------------------------------|

Definition at line 125 of file `complex`.

## 4.662.2 Member Typedef Documentation

4.662.2.1 `template<typename _Tp> typedef _Tp std::complex< _Tp >::value_type`

Value typedef.

Definition at line 128 of file `complex`.

## 4.662.3 Constructor &amp; Destructor Documentation

4.662.3.1 `template<typename _Tp> constexpr std::complex< _Tp >::complex ( const _Tp & _r = _Tp(), const _Tp & _i = _Tp() ) [inline]`

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

Definition at line 132 of file `complex`.

4.662.3.2 `template<typename _Tp> template<typename _Up> constexpr std::complex< _Tp >::complex ( const complex< _Up> & _z ) [inline]`

Copy constructor.

Definition at line 139 of file `complex`.

## 4.662.4 Member Function Documentation

4.662.4.1 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator+=( const _Tp & _t ) [inline]`

Add *t* to this complex number.

Definition at line 184 of file `complex`.

4.662.4.2 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator-= ( const _Tp & _t ) [inline]`

Subtract *t* from this complex number.

Definition at line 193 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

4.663 `std::complex< double >` Struct Template Reference

## Public Types

- typedef `__complex__ double` **\_ComplexT**
- typedef `double` **value\_type**

## Public Member Functions

- constexpr **complex** (`_ComplexT __z`)
- constexpr **complex** (`double __r=0.0, double __i=0.0`)
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr `double` **real**()
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr `double` **imag**()
- constexpr `_ComplexT` **\_\_rep** () const
- void **imag** (`double __val`)
- **complex** & **operator\*=** (`double __d`)
- template<typename `_Tp` >  
**complex** & **operator\*=** (`const complex<_Tp> &__z`)
- **complex** & **operator+=** (`double __d`)
- template<typename `_Tp` >  
**complex** & **operator+=** (`const complex<_Tp> &__z`)
- **complex** & **operator-=** (`double __d`)
- template<typename `_Tp` >  
**complex** & **operator-=** (`const complex<_Tp> &__z`)
- **complex** & **operator/=** (`double __d`)
- template<typename `_Tp` >  
**complex** & **operator/=** (`const complex<_Tp> &__z`)
- **complex** & **operator=** (`double __d`)
- template<typename `_Tp` >  
**complex** & **operator=** (`const complex<_Tp> &__z`)
- void **real** (`double __val`)

## 4.663.1 Detailed Description

template<>struct `std::complex< double >`

26.2.3 complex specializations `complex<double>` specialization

Definition at line 1193 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

4.664 `std::complex< float >` Struct Template Reference

## Public Types

- typedef `__complex__ float` **\_ComplexT**
- typedef `float` **value\_type**

## Public Member Functions

- constexpr **complex** (`_ComplexT __z`)
- constexpr **complex** (`float __r=0.0f, float __i=0.0f`)
- constexpr **complex** (`const complex< long double > &`)
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr float `real()`
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr float `imag()`
- constexpr `_ComplexT __rep` () const
- void **imag** (`float __val`)
- **complex** & **operator\*=** (`float __f`)
- template<class `_Tp` >  
**complex** & **operator\*=** (`const complex< _Tp > &__z`)
- **complex** & **operator+=** (`float __f`)
- template<typename `_Tp` >  
**complex** & **operator+=** (`const complex< _Tp > &__z`)
- **complex** & **operator-=** (`float __f`)
- template<class `_Tp` >  
**complex** & **operator-=** (`const complex< _Tp > &__z`)
- **complex** & **operator/=** (`float __f`)
- template<class `_Tp` >  
**complex** & **operator/=** (`const complex< _Tp > &__z`)
- **complex** & **operator=** (`float __f`)
- template<typename `_Tp` >  
**complex** & **operator=** (`const complex< _Tp > &__z`)
- void **real** (`float __val`)

## 4.664.1 Detailed Description

template<>struct std::complex< float >

26.2.3 complex specializations complex<float> specialization

Definition at line 1044 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

4.665 `std::complex< long double >` Struct Template Reference

## Public Types

- typedef `__complex__ long double` **\_ComplexT**
- typedef `long double` **value\_type**

## Public Member Functions

- constexpr **complex** (`_ComplexT __z`)
- constexpr **complex** (`long double __r=0.0L, long double __i=0.0L`)

## 4.665.1 Detailed Description

template<> struct std::complex< long double >

26.2.3 complex specializations complex<long double> specialization

Definition at line 1343 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

## 4.666 std::condition\_variable Class Reference

## Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

## Public Member Functions

- **condition\_variable** (const [condition\\_variable](#) &)=delete
- native\_handle\_type **native\_handle** ()
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable](#) & **operator=** (const [condition\\_variable](#) &)=delete
- void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock)
- template<typename \_Predicate >  
void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, \_Predicate \_\_p)
- template<typename \_Rep, typename \_Period >  
[cv\\_status](#) **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Rep, typename \_Period, typename \_Predicate >  
bool **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Duration >  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_\_clock\_t, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration >  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration, typename \_Predicate >  
bool **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

## 4.666.1 Detailed Description

condition\_variable

Definition at line 59 of file condition\_variable.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 4.667 std::condition\_variable\_any Class Reference

## Public Member Functions

- **condition\_variable\_any** (const [condition\\_variable\\_any](#) &)=delete
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable\\_any](#) & **operator=** (const [condition\\_variable\\_any](#) &)=delete
- template<typename \_Lock >  
void **wait** (\_Lock &\_\_lock)
- template<typename \_Lock, typename \_Predicate >  
void **wait** (\_Lock &\_\_lock, \_Predicate \_\_p)
- template<typename \_Lock, typename \_Rep, typename \_Period >  
[cv\\_status](#) **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Lock, typename \_Rep, typename \_Period, typename \_Predicate >  
bool **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Lock, typename \_Clock, typename \_Duration >  
[cv\\_status](#) **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Lock, typename \_Clock, typename \_Duration, typename \_Predicate >  
bool **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

## 4.667.1 Detailed Description

`condition_variable_any`

Definition at line 170 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 4.668 std::conditional&lt; \_Cond, \_Iftrue, \_Iffalse &gt; Struct Template Reference

## Public Types

- typedef \_Iftrue **type**

## 4.668.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse> struct std::conditional< _Cond, _Iftrue, _Iffalse >
```

Define a member typedef `type` to one of two argument types.

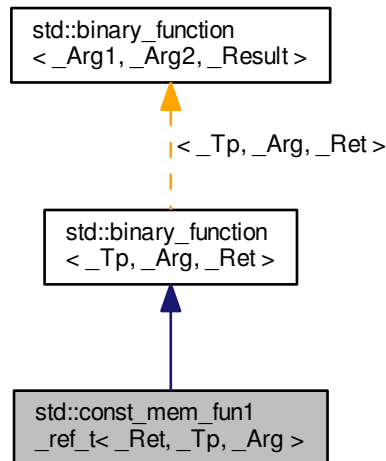
Definition at line 1780 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.669 std::const\_mem\_fun1\_ref\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- **const\_mem\_fun1\_ref\_t** (`_Ret(_Tp::*__pf)(_Arg) const`)
- `_Ret` **operator()** (`(const _Tp &__r, _Arg __x) const`)

## 4.669.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 668 of file `stl_function.h`.

## 4.669.2 Member Typedef Documentation

4.669.2.1 typedef `_Tp` `std::binary_function<_Tp, _Arg, _Ret>::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.669.2.2 typedef \_Ret std::binary\_function< \_Tp, \_Arg, \_Ret >::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

4.669.2.3 typedef \_Arg std::binary\_function< \_Tp, \_Arg, \_Ret >::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

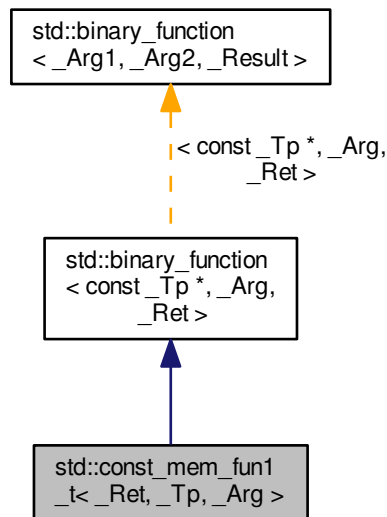
Definition at line 120 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.670 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

Inheritance diagram for std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef const \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

### Public Member Functions

- **const\_mem\_fun1\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg) const)

- `_Ret operator()` (`const _Tp *__p, _Arg __x`) `const`

#### 4.670.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg>class std::const_mem_fun1_t<_Ret, _Tp, _Arg>`

One of the [adaptors for member pointers](#).

Definition at line 632 of file `stl_function.h`.

#### 4.670.2 Member Typedef Documentation

4.670.2.1 `typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.670.2.2 `typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.670.2.3 `typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

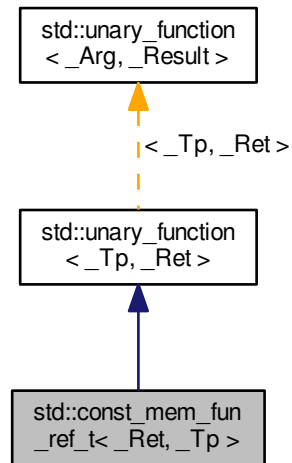
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.671 std::const\_mem\_fun\_ref\_t&lt; \_Ret, \_Tp &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >:



## Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

## Public Member Functions

- **const\_mem\_fun\_ref\_t** (\_Ret(\_Tp::\* \_\_pf)() const)
- \_Ret **operator()** (const \_Tp &\_\_r) const

## 4.671.1 Detailed Description

```
template<typename _Ret, typename _Tp>class std::const_mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

Definition at line 596 of file stl\_function.h.

## 4.671.2 Member Typedef Documentation

4.671.2.1 typedef \_Tp std::unary\_function< \_Tp, \_Ret >::argument\_type [inherited]

argument\_type is the type of the argument

Definition at line 104 of file stl\_function.h.

4.671.2.2 `typedef _Ret std::unary_function<_Tp, _Ret>::result_type` [inherited]

`result_type` is the return type

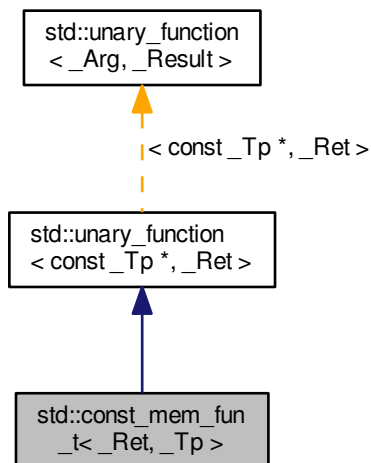
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

4.672 `std::const_mem_fun_t<_Ret, _Tp>` Class Template Reference

Inheritance diagram for `std::const_mem_fun_t<_Ret, _Tp>`:



## Public Types

- `typedef const _Tp *` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

## Public Member Functions

- `const_mem_fun_t(_Ret(_Tp::*__pf)()) const`
- `_Ret operator() (const _Tp *__p) const`

## 4.672.1 Detailed Description

`template<typename _Ret, typename _Tp>class std::const_mem_fun_t<_Ret, _Tp>`

One of the [adaptors for member pointers](#).

Definition at line 560 of file stl\_function.h.

#### 4.672.2 Member Typedef Documentation

4.672.2.1 `typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl\_function.h.

4.672.2.2 `typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type` [inherited]

`result_type` is the return type

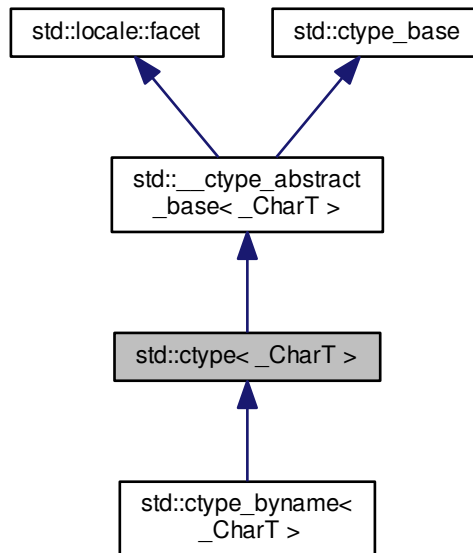
Definition at line 107 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.673 std::ctype<\_CharT> Class Template Reference

Inheritance diagram for `std::ctype<_CharT>`:



#### Public Types

- `typedef const int * __to_type`
- `typedef _CharT char_type`

- typedef [\\_\\_ctype\\_abstract\\_base](#)  
 <\_CharT>::mask **mask**

### Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- bool [is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const [char\\_type](#) \* [scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual char [do\\_narrow](#) ([char\\_type](#), char \_\_dfault) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_dest) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.673.1 Detailed Description

template<typename \_CharT>class std::ctype< \_CharT >

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in \_\_ctype\_abstract\_base, to allow for implementation flexibility. See ctype<wchar\_t> for an example. The functions are documented in \_\_ctype\_abstract\_base.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 605 of file locale\_facets.h.

## 4.673.2 Member Function Documentation

4.673.2.1 template<typename \_CharT> virtual bool std::ctype< \_CharT >::do\_is ( mask \_\_m, char\_type \_\_c ) const  
[protected], [virtual]

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

## Returns

(M & \_\_m) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

4.673.2.2 template<typename \_CharT> virtual const char\_type\* std::ctype< \_CharT >::do\_is ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec ) const [protected], [virtual]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.673.2.3** `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow ( char_type __c, char __dfault ) const [protected], [virtual]`

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::narrow()`.

**4.673.2.4** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [protected], [virtual]`

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**`__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

4.673.2.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**Pointer to a matching char\_type if found, else `__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

4.673.2.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [`lo`,`hi`) for which `is(m,c)` is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else *\_\_hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.673.2.7** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower ( char_type __c ) const`  
`[protected], [virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|            |                           |
|------------|---------------------------|
| <i>__c</i> | The char_type to convert. |
|------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else *\_\_c*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::tolower()`.

**4.673.2.8** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo,`  
`const char_type * __hi ) const` `[protected], [virtual]`

Convert array to lowercase.

This virtual function converts each char\_type in the range [*\_\_lo*,*\_\_hi*) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|             |                            |
|-------------|----------------------------|
| <i>__lo</i> | Pointer to start of range. |
| <i>__hi</i> | Pointer to end of range.   |

**Returns**

*\_\_hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.673.2.9** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type __c ) const`  
`[protected], [virtual]`

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

#### Returns

The uppercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::toupper()`.

**4.673.2.10** `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected], [virtual]`

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**4.673.2.11** `template<typename _CharT> virtual char_type std::ctype< _CharT >::do_widen ( char __c ) const [protected], [virtual]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by std::ctype< char >::widen().

**4.673.2.12** `template<typename _CharT> virtual const char* std::ctype< _CharT >::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` [protected],[virtual]

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start range.           |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**4.673.2.13** `template<typename _CharT> bool std::__ctype_abstract_base< _CharT >::is ( mask __m, char_type __c ) const` [inline],[inherited]

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

#### Returns

$(M \& \text{__m}) \neq 0$ .

Definition at line 162 of file locale\_facets.h.

Referenced by std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**4.673.2.14** `template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` [inline],[inherited]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**`__hi`.

Definition at line 179 of file locale\_facets.h.

4.673.2.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow ( char_type __c, char __dfault ) const [inline],[inherited]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

**Returns**The converted `char`.

Definition at line 324 of file locale\_facets.h.

Referenced by `std::time_put<_CharT, _OutIter>::put()`.

4.673.2.16 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline],[inherited]`

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**`__hi`.

Definition at line 346 of file locale\_facets.h.

4.673.2.17 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline],[inherited]`

Find `char_type` matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to matching char\_type if found, else `__hi`.

Definition at line 195 of file locale\_facets.h.

**4.673.2.18** `template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline],[inherited]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

#### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

#### Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale\_facets.h.

**4.673.2.19** `template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::tolower ( char_type __c ) const` `[inline],[inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

#### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

#### Returns

The lowercase char\_type if convertible, else `__c`.

Definition at line 254 of file locale\_facets.h.

**4.673.2.20** `template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::tolower ( char_type * __lo, const char_type * __hi ) const` `[inline],[inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo,__hi)`.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Definition at line 269 of file `locale_facets.h`.

**4.673.2.21** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline],[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

#### Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

**4.673.2.22** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline],[inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Definition at line 240 of file `locale_facets.h`.

**4.673.2.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline],[inherited]`

Widen char to `char_type`.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted char\_type.

Definition at line 286 of file locale\_facets.h.

Referenced by std::money\_get<\_CharT, \_InIter>::do\_get(), std::time\_put<\_CharT, \_OutIter>::do\_put(), std::money\_put<\_CharT, \_OutIter>::do\_put(), std::tr2::operator<<(), and std::operator<<().

**4.673.2.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline],[inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 305 of file locale\_facets.h.

### 4.673.3 Member Data Documentation

**4.673.3.1** `template<typename _CharT> locale::id std::ctype<_CharT>::id [static]`

The facet id for ctype<char\_type>

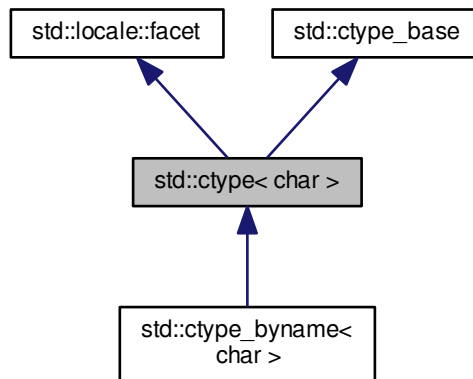
Definition at line 613 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.674 std::ctype&lt; char &gt; Class Template Reference

Inheritance diagram for std::ctype< char >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef char **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- **ctype** (const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- **ctype** (\_\_c\_locale \_\_cloc, const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char \_\_c) const
- const char \* **is** (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const char \* **scan\_is** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* **scan\_not** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* **table** () const throw ()
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Public Member Functions

- static const mask \* **classic\_table** () throw ()

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual [~ctype](#) ()
- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_ctype](#)
- bool [\\_M\\_del](#)
- char [\\_M\\_narrow](#) [1+static\_cast< unsigned char >(-1)]
- char [\\_M\\_narrow\\_ok](#)
- const mask \* [\\_M\\_table](#)
- \_\_to\_type [\\_M\\_tolower](#)
- \_\_to\_type [\\_M\\_toupper](#)
- char [\\_M\\_widen](#) [1+static\_cast< unsigned char >(-1)]
- char [\\_M\\_widen\\_ok](#)

## 4.674.1 Detailed Description

template<>class std::ctype< char >

The ctype<char> specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 674 of file locale\_facets.h.

## 4.674.2 Member Typedef Documentation

## 4.674.2.1 typedef char std::ctype&lt; char &gt;::char\_type

Typedef for the template parameter char.

Definition at line 679 of file locale\_facets.h.

## 4.674.3 Constructor &amp; Destructor Documentation

## 4.674.3.1 std::ctype&lt; char &gt;::ctype ( const mask \* \_\_table = 0, bool \_\_del = false, size\_t \_\_refs = 0 ) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| <code>__table</code> | If non-zero, table is used as the per-char mask. Else classic_table() is used. |
| <code>__del</code>   | If true, passes ownership of table to this facet.                              |
| <code>__refs</code>  | Passed to the base facet class.                                                |

## 4.674.3.2 std::ctype&lt; char &gt;::ctype ( \_\_c\_locale \_\_cloc, const mask \* \_\_table = 0, bool \_\_del = false, size\_t \_\_refs = 0 ) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__cloc</code>  | Handle to C locale data.                          |
| <code>__table</code> | If non-zero, table is used as the per-char mask.  |
| <code>__del</code>   | If true, passes ownership of table to this facet. |
| <code>__refs</code>  | Passed to the base facet class.                   |

## 4.674.3.3 virtual std::ctype&lt; char &gt;::~~ctype ( ) [protected],[virtual]

Destructor.

This function deletes table() if del was true in the constructor.

## 4.674.4 Member Function Documentation

4.674.4.1 static const mask\* std::ctype< char >::classic\_table ( ) throw () [static]

Returns a pointer to the C locale mask table.

4.674.4.2 virtual char std::ctype< char >::do\_narrow ( char\_type \_\_c, char \_\_dfault ) const [inline],[protected],[virtual]

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Definition at line 1124 of file locale\_facets.h.

4.674.4.3 virtual const char\_type\* std::ctype< char >::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to ) const [inline],[protected],[virtual]

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, dfault is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 1150 of file locale\_facets.h.

4.674.4.4 virtual char\_type std::ctype< char >::do\_tolower ( char\_type \_\_c ) const [protected],[virtual]

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The lowercase char if convertible, else `__c`.

**4.674.4.5** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

#### Returns

`__hi`.

**4.674.4.6** `virtual char_type std::ctype< char >::do_toupper ( char_type __c ) const` `[protected]`, `[virtual]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The uppercase char if convertible, else `__c`.

**4.674.4.7** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

**4.674.4.8** `virtual char_type std::ctype< char >::do_widen ( char __c ) const` `[inline], [protected], [virtual]`

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted character.

Definition at line 1075 of file locale\_facets.h.

**4.674.4.9** `virtual const char* std::ctype< char >::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[inline], [protected], [virtual]`

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

**Returns**`__hi`.

Definition at line 1098 of file locale\_facets.h.

**4.674.4.10** `bool std::ctype< char >::is ( mask __m, char __c ) const` `[inline]`

Test char classification.

This function compares the mask table[c] to `__m`.**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The char to compare the mask of. |
| <code>__m</code> | The mask to compare against.     |

**Returns**True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file ctype\_inline.h.

**4.674.4.11** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const` `[inline]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**`__hi`.

Definition at line 48 of file ctype\_inline.h.

**4.674.4.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault ) const` `[inline]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype&lt;char&gt; facet, c will be returned unchanged.

This function works as if it returns ctype&lt;char&gt;::do\_narrow(c). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

**Returns**

The converted character.

Definition at line 923 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_narrow().

**4.674.4.13** `const char_type* std::ctype< char >::narrow ( const char_type * __lo, const char_type * __hi, char __default, char * __to ) const` [inline]

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, default, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 956 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_narrow().

**4.674.4.14** `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const` [inline]

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file ctype\_inline.h.

**4.674.4.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const` [inline]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [\_\_lo,\_\_hi) for which is(m,char) is false.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file `ctype_inline.h`.

**4.674.4.16** `const mask* std::ctype< char >::table ( ) const throw () [inline]`

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

Definition at line 974 of file `locale_facets.h`.

**4.674.4.17** `char_type std::ctype< char >::tolower ( char_type __c ) const [inline]`

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

## Returns

The lowercase char if convertible, else `__c`.

Definition at line 828 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_tolower()`.

**4.674.4.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo, const char_type * __hi ) const [inline]`

Convert array to lowercase.

This function converts each char in the range `[lo,hi)` to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

## Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

## Returns

`__hi`.

Definition at line 845 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_tolower()`.

**4.674.4.19 char\_type std::ctype< char >::toupper ( char\_type \_\_c ) const** [inline]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do\_toupper(c). do\_toupper() must always return the same result for the same input.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The uppercase char if convertible, else `__c`.

Definition at line 795 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

**4.674.4.20 const char\_type\* std::ctype< char >::toupper ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const** [inline]

Convert array to uppercase.

This function converts each char in the range [`__lo`,`__hi`) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>::do\_toupper(`__lo`, `__hi`). do\_toupper() must always return the same result for the same input.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

`__hi`.

Definition at line 812 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

**4.674.4.21 char\_type std::ctype< char >::widen ( char \_\_c ) const** [inline]

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do\_widen(c). do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted character.

Definition at line 865 of file locale\_facets.h.

References `std::ctype< _CharT >::do_widen()`.

**4.674.4.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const`  
`[inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to first char in range.   |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 892 of file locale\_facets.h.

References `std::ctype< _CharT >::do_widen()`.

**4.674.5 Member Data Documentation**

**4.674.5.1** `locale::id std::ctype< char >::id` `[static]`

The facet id for `ctype<char>`

Definition at line 696 of file locale\_facets.h.

**4.674.5.2** `const size_t std::ctype< char >::table_size` `[static]`

The size of the mask table. It is `SCHAR_MAX + 1`.

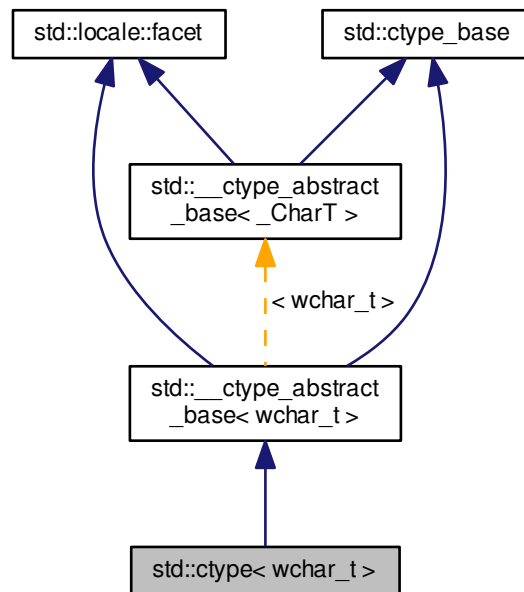
Definition at line 698 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

## 4.675 std::ctype&lt; wchar\_t &gt; Class Template Reference

Inheritance diagram for std::ctype< wchar\_t >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef wctype\_t **\_\_wmask\_type**
- typedef wchar\_t **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- **ctype** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- const **char\_type** \* **is** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (**char\_type** \_\_c, char \_\_default) const
- const **char\_type** \* **narrow** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, char \_\_default, char \* \_\_to) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const **char\_type** \* **scan\_not** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **tolower** (**char\_type** \_\_c) const
- const **char\_type** \* **tolower** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **toupper** (**char\_type** \_\_c) const
- const **char\_type** \* **toupper** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const

- [char\\_type widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

#### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

#### Protected Member Functions

- virtual [~ctype](#) ()
- [\\_\\_wmask\\_type](#) **\_M\_convert\_to\_wmask** (const mask \_\_m) const throw ()
- void **\_M\_initialize\_ctype** () throw ()
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual char [do\\_narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

#### Static Protected Member Functions

- static [\\_\\_c\\_locale](#) **\_S\_clone\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc, const char \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void **\_S\_destroy\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale](#) **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static [\\_\\_c\\_locale](#) **\_S\_lc\_ctype\_c\_locale** ([\\_\\_c\\_locale](#) \_\_cloc, const char \* \_\_s)

## Protected Attributes

- mask **\_M\_bit** [16]
- \_\_c\_locale **\_M\_c\_locale\_ctype**
- char **\_M\_narrow** [128]
- bool **\_M\_narrow\_ok**
- wint\_t **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- \_\_wmask\_type **\_M\_wmask** [16]

## 4.675.1 Detailed Description

template<>class std::ctype< wchar\_t >

The ctype<wchar\_t> specialization.

This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.

ctype<wchar\_t> inherits its public methods from \_\_ctype\_abstract\_base<wchar\_t>.

Definition at line 1175 of file locale\_facets.h.

## 4.675.2 Member Typedef Documentation

## 4.675.2.1 typedef wchar\_t std::ctype&lt; wchar\_t &gt;::char\_type

Typedef for the template parameter wchar\_t.

Definition at line 1180 of file locale\_facets.h.

## 4.675.3 Constructor &amp; Destructor Documentation

## 4.675.3.1 std::ctype&lt; wchar\_t &gt;::ctype ( size\_t \_\_refs = 0 ) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|               |                                 |
|---------------|---------------------------------|
| <b>__refs</b> | Passed to the base facet class. |
|---------------|---------------------------------|

## 4.675.3.2 std::ctype&lt; wchar\_t &gt;::ctype ( \_\_c\_locale \_\_cloc, size\_t \_\_refs = 0 ) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

## Parameters

|               |                                 |
|---------------|---------------------------------|
| <b>__cloc</b> | Handle to C locale data.        |
| <b>__refs</b> | Passed to the base facet class. |

4.675.3.3 virtual std::ctype< wchar\_t >::~~ctype ( ) [protected], [virtual]

Destructor.

#### 4.675.4 Member Function Documentation

4.675.4.1 virtual bool std::ctype< wchar\_t >::do\_is ( mask \_\_m, char\_type \_\_c ) const [protected], [virtual]

Test wchar\_t classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

##### Parameters

|     |                                  |
|-----|----------------------------------|
| __c | The wchar_t to find the mask of. |
| __m | The mask to compare against.     |

##### Returns

(M & \_\_m) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

4.675.4.2 virtual const char\_type\* std::ctype< wchar\_t >::do\_is ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec ) const [protected], [virtual]

Return a mask array.

This function finds the mask for each wchar\_t in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

##### Parameters

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

##### Returns

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

4.675.4.3 virtual char std::ctype< wchar\_t >::do\_narrow ( char\_type \_\_c, char \_\_dfault ) const [protected], [virtual]

Narrow wchar\_t to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<wchar\_t> facet, c will be cast to char and returned.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                      |
|-----------------------|--------------------------------------|
| <code>__c</code>      | The <code>wchar_t</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.  |

#### Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.4** `virtual const char_type* std::ctype< wchar_t >::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` [protected], [virtual]

Narrow `wchar_t` array to char array.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `wchar_t` in the input that cannot be converted, `dfault` is used instead. For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to `char`.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.5** `virtual const char_type* std::ctype< wchar_t >::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` [protected], [virtual]

Find `wchar_t` matching mask.

This function searches for and returns the first `wchar_t` `c` in `[__lo,__hi)` for which `is(__m,c)` is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching wchar\_t if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.6** virtual const char\_type\* std::ctype< wchar\_t >::do\_scan\_not ( mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi ) const [protected], [virtual]

Find wchar\_t not matching mask.

This function searches for and returns a pointer to the first wchar\_t c of [ \_\_lo, \_\_hi) for which is(\_\_m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching wchar\_t if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.7** virtual char\_type std::ctype< wchar\_t >::do\_tolower ( char\_type \_\_c ) const [protected], [virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The wchar_t to convert. |
|------------------|-------------------------|

**Returns**

The lowercase wchar\_t if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.8** virtual const char\_type\* std::ctype< wchar\_t >::do\_tolower ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const [protected], [virtual]

Convert array to lowercase.

This virtual function converts each wchar\_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.9** `virtual char_type std::ctype< wchar_t >::do_toupper ( char_type __c ) const` `[protected]`, `[virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__c</code> | The <code>wchar_t</code> to convert. |
|------------------|--------------------------------------|

## Returns

The uppercase `wchar_t` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.10** `virtual const char_type* std::ctype< wchar_t >::do_toupper ( char_type * __lo, const char_type * __hi ) const` `[protected]`, `[virtual]`

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.11** `virtual char_type std::ctype< wchar_t >::do_widen ( char __c ) const` `[protected]`, `[virtual]`

Widen `char` to `wchar_t`.

This virtual function converts the `char` to `wchar_t` using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be cast to `wchar_t`.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted wchar\_t.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.12** virtual const char\* **std::ctype< wchar\_t >::do\_widen** ( const char \* *\_\_lo*, const char \* *\_\_hi*, char\_type \* *\_\_to* ) const  
[protected], [virtual]

Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be copied, casting each element to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start range.           |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.675.4.13** bool **std::\_\_ctype\_abstract\_base< wchar\_t >::is** ( mask *\_\_m*, char\_type *\_\_c* ) const  
[inline], [inherited]

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

#### Returns

(M & `__m`) != 0.

Definition at line 162 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_is().

**4.675.4.14** `const char_type* std::__ctype_abstract_base< wchar_t >::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` [inline],[inherited]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Definition at line 179 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_is().

**4.675.4.15** `char std::__ctype_abstract_base< wchar_t >::narrow ( char_type __c, char __dfault ) const` [inline],[inherited]

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Definition at line 324 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_narrow().

**4.675.4.16** `const char_type* std::__ctype_abstract_base< wchar_t >::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const` [inline],[inherited]

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_dfault, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 346 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_narrow().

**4.675.4.17** `const char_type* std::__ctype_abstract_base< wchar_t >::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` [inline],[inherited]

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to matching char\_type if found, else `__hi`.

Definition at line 195 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_scan\_is().

**4.675.4.18** `const char_type* std::__ctype_abstract_base< wchar_t >::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` [inline],[inherited]

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

## Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

## Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_scan\_not().

**4.675.4.19** `char_type std::__ctype_abstract_base< wchar_t >::tolower ( char_type __c ) const` `[inline]`,  
`[inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

#### Returns

The lowercase `char_type` if convertible, else `__c`.

Definition at line 254 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_tolower()`.

**4.675.4.20** `const char_type* std::__ctype_abstract_base< wchar_t >::tolower ( char_type * __lo, const char_type * __hi ) const` `[inline]`,  
`[inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Definition at line 269 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_tolower()`.

**4.675.4.21** `char_type std::__ctype_abstract_base< wchar_t >::toupper ( char_type __c ) const` `[inline]`,  
`[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

#### Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_toupper().

**4.675.4.22** `const char_type* std::__ctype_abstract_base< wchar_t >::toupper ( char_type * __lo, const char_type * __hi ) const` [inline],[inherited]

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Definition at line 240 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_toupper().

**4.675.4.23** `char_type std::__ctype_abstract_base< wchar_t >::widen ( char __c ) const` [inline],[inherited]

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted char\_type.

Definition at line 286 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_widen().

**4.675.4.24** `const char* std::__ctype_abstract_base< wchar_t >::widen ( const char * __lo, const char * __hi, char_type * __to ) const` [inline],[inherited]

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 305 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

## 4.675.5 Member Data Documentation

## 4.675.5.1 locale::id std::ctype&lt;wchar\_t&gt;::id [static]

The facet id for `ctype<wchar_t>`

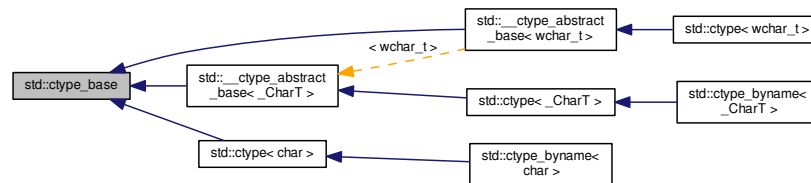
Definition at line 1198 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.676 std::ctype\_base Struct Reference

Inheritance diagram for `std::ctype_base`:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## 4.676.1 Detailed Description

Base class for ctype.

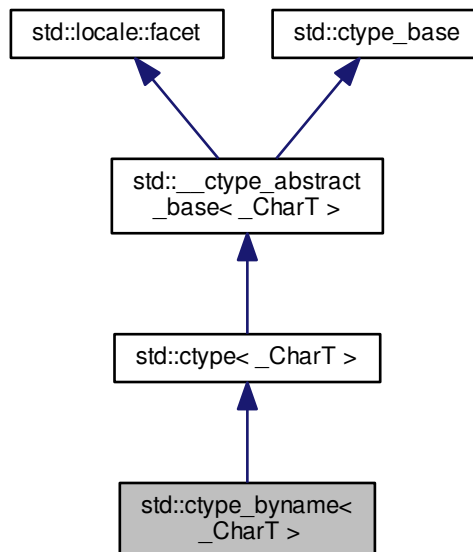
Definition at line 41 of file ctype\_base.h.

The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

## 4.677 std::ctype\_byname&lt; \_CharT &gt; Class Template Reference

Inheritance diagram for std::ctype\_byname< \_CharT >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef `ctype< _CharT >::mask` **mask**

## Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, `char_type` \_\_c) const
- const `char_type` \* **is** (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (`char_type` \_\_c, char \_\_dfault) const

- const `char_type` \* `narrow` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, char \_\_default, char \* \_\_to) const
- const `char_type` \* `scan_is` (mask \_\_m, const `char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- const `char_type` \* `scan_not` (mask \_\_m, const `char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- `char_type` `tolower` (`char_type` \_\_c) const
- const `char_type` \* `tolower` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- `char_type` `toupper` (`char_type` \_\_c) const
- const `char_type` \* `toupper` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- `char_type` `widen` (char \_\_c) const
- const char \* `widen` (const char \* \_\_lo, const char \* \_\_hi, `char_type` \* \_\_to) const

#### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static `locale::id` **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

#### Protected Member Functions

- virtual bool `do_is` (mask \_\_m, `char_type` \_\_c) const
- virtual const `char_type` \* `do_is` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, mask \* \_\_vec) const
- virtual char `do_narrow` (`char_type`, char \_\_default) const
- virtual const `char_type` \* `do_narrow` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, char \_\_default, char \* \_\_to) const
- virtual const `char_type` \* `do_scan_is` (mask \_\_m, const `char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- virtual const `char_type` \* `do_scan_not` (mask \_\_m, const `char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- virtual `char_type` `do_tolower` (`char_type` \_\_c) const
- virtual const `char_type` \* `do_tolower` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- virtual `char_type` `do_toupper` (`char_type` \_\_c) const
- virtual const `char_type` \* `do_toupper` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const
- virtual `char_type` `do_widen` (char \_\_c) const
- virtual const char \* `do_widen` (const char \* \_\_lo, const char \* \_\_hi, `char_type` \* \_\_dest) const

#### Static Protected Member Functions

- static `_c_locale` **\_S\_clone\_c\_locale** (`_c_locale` & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (`_c_locale` & \_\_cloc, const char \* \_\_s, `_c_locale` \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (`_c_locale` & \_\_cloc)
- static `_c_locale` **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static `_c_locale` **\_S\_lc\_ctype\_c\_locale** (`_c_locale` \_\_cloc, const char \* \_\_s)

## 4.677.1 Detailed Description

```
template<typename _CharT> class std::ctype_byname< _CharT >
```

class ctype\_byname [22.2.1.2].

Definition at line 1467 of file locale\_facets.h.

## 4.677.2 Member Function Documentation

4.677.2.1 `template<typename _CharT> virtual bool std::ctype< _CharT >::do_is ( mask __m, char_type __c ) const` [protected], [virtual], [inherited]

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

## Returns

$(M \& \text{__m}) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

4.677.2.2 `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` [protected], [virtual], [inherited]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**`__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.3** `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow ( char_type __c, char __default ) const` [protected], [virtual], [inherited]

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `__default` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**The converted `char`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).Referenced by `std::ctype<char>::narrow()`.

**4.677.2.4** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow ( const char_type * __lo, const char_type * __hi, char __default, char * __to ) const` [protected], [virtual], [inherited]

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**`__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.5** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.6** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.7** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower ( char_type __c ) const` [protected], [virtual], [inherited]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by std::ctype<char>::tolower().

**4.677.2.8** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Convert array to lowercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.9** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type __c ) const` [protected], [virtual], [inherited]

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by std::ctype<char>::toupper().

**4.677.2.10** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper ( char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.11** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen ( char __c ) const`  
`[protected], [virtual], [inherited]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::widen()`.

**4.677.2.12** `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const`  
`[protected], [virtual], [inherited]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start range.           |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

**Returns**`__hi.`Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.677.2.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c )  
const [inline],[inherited]`

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

**Returns**`(M & __m) != 0.`Definition at line 162 of file `locale_facets.h`.Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**4.677.2.14** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is ( const char_type  
* __lo, const char_type * __hi, mask * __vec ) const [inline],[inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**`__hi.`Definition at line 179 of file `locale_facets.h`.

**4.677.2.15** `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow ( char_type __c, char  
__dfault ) const [inline],[inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

## Returns

The converted char.

Definition at line 324 of file locale\_facets.h.

Referenced by std::time\_put<\_CharT, \_OutIter>::put().

**4.677.2.16** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const char_type * __lo, const char_type * __hi, char __default, char * __to ) const` [inline],[inherited]

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *default* is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 346 of file locale\_facets.h.

**4.677.2.17** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` [inline],[inherited]

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to matching char\_type if found, else `__hi`.

Definition at line 195 of file locale\_facets.h.

**4.677.2.18** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` `[inline],[inherited]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

#### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

#### Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale\_facets.h.

**4.677.2.19** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const` `[inline],[inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

#### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

#### Returns

The lowercase char\_type if convertible, else `__c`.

Definition at line 254 of file locale\_facets.h.

**4.677.2.20** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower ( char_type * __lo, const char_type * __hi ) const` `[inline],[inherited]`

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo,\_\_hi).

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Definition at line 269 of file locale\_facets.h.

4.677.2.21 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c )  
const [inline],[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

#### Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

4.677.2.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type  
* __lo, const char_type * __hi ) const [inline],[inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Definition at line 240 of file `locale_facets.h`.

4.677.2.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const  
[inline],[inherited]`

Widen char to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted `char_type`.

Definition at line 286 of file `locale_facets.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::tr2::operator<<()`, and `std::operator<<()`.

**4.677.2.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[inline], [inherited]`

Widen array to `char_type`.

This function converts each `char` in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 305 of file `locale_facets.h`.

### 4.677.3 Member Data Documentation

**4.677.3.1** `template<typename _CharT> locale::id std::ctype<_CharT>::id` `[static], [inherited]`

The facet id for `ctype<char_type>`

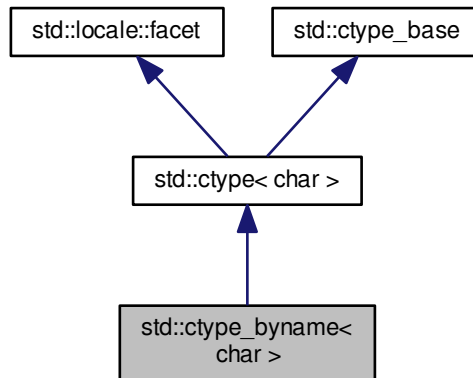
Definition at line 613 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.678 std::ctype\_byname&lt; char &gt; Class Template Reference

Inheritance diagram for std::ctype\_byname< char >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef char **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char \_\_c) const
- const char \* **is** (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* **scan\_is** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* **scan\_not** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* **table** () const throw ()
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Public Member Functions

- static const mask \* **classic\_table** () throw ()

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_ctype**
- bool **\_M\_del**
- char **\_M\_narrow** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_narrow\_ok**
- const mask \* **\_M\_table**
- \_\_to\_type **\_M\_tolower**
- \_\_to\_type **\_M\_toupper**
- char **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_widen\_ok**

## 4.678.1 Detailed Description

template<>class std::ctype\_byname< char >

22.2.1.4 Class ctype\_byname specializations.

Definition at line 1482 of file locale\_facets.h.

## 4.678.2 Member Typedef Documentation

4.678.2.1 typedef char std::ctype< char >::char\_type [inherited]

Typedef for the template parameter char.

Definition at line 679 of file locale\_facets.h.

## 4.678.3 Member Function Documentation

4.678.3.1 static const mask\* std::ctype< char >::classic\_table ( ) throw () [static],[inherited]

Returns a pointer to the C locale mask table.

4.678.3.2 virtual char std::ctype< char >::do\_narrow ( char\_type \_\_c, char \_\_dfault ) const [inline],[protected],[virtual],[inherited]

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

## Returns

The converted char.

Definition at line 1124 of file locale\_facets.h.

4.678.3.3 virtual const char\_type\* std::ctype< char >::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to ) const [inline],[protected],[virtual],[inherited]

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, dfault is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 1150 of file locale\_facets.h.

**4.678.3.4** `virtual char_type std::ctype< char >::do_tolower ( char_type __c ) const` [protected], [virtual], [inherited]

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The lowercase char if convertible, else `__c`.

**4.678.3.5** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

#### Returns

`__hi`.

**4.678.3.6** `virtual char_type std::ctype< char >::do_toupper ( char_type __c ) const` [protected], [virtual], [inherited]

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The uppercase char if convertible, else `__c`.

**4.678.3.7** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const` [protected], [virtual], [inherited]

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

**4.678.3.8** `virtual char_type std::ctype< char >::do_widen ( char __c ) const` [inline], [protected], [virtual], [inherited]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted character.

Definition at line 1075 of file locale\_facets.h.

**4.678.3.9** `virtual const char* std::ctype< char >::do_widen ( const char * __lo, const char * __hi, char_type * __to ) const`  
`[inline], [protected], [virtual], [inherited]`

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 1098 of file locale\_facets.h.

**4.678.3.10** `bool std::ctype< char >::is ( mask __m, char __c ) const` `[inline], [inherited]`

Test char classification.

This function compares the mask table[c] to `__m`.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The char to compare the mask of. |
| <code>__m</code> | The mask to compare against.     |

#### Returns

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file ctype\_inline.h.

**4.678.3.11** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const` `[inline], [inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns***\_\_hi*.

Definition at line 48 of file ctype\_inline.h.

**4.678.3.12** char std::ctype< char >::narrow ( char\_type \_\_c, char \_\_dfault ) const [inline],[inherited]

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

This function works as if it returns ctype<char>::do\_narrow(c). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                 |                                     |
|-----------------|-------------------------------------|
| <i>__c</i>      | The char to convert.                |
| <i>__dfault</i> | Char to return if conversion fails. |

**Returns**

The converted character.

Definition at line 923 of file locale\_facets.h.

References std::ctype&lt; \_CharT &gt;::do\_narrow().

**4.678.3.13** const char\_type\* std::ctype< char >::narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to ) const [inline],[inherited]

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, dfault is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, dfault, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                 |                                   |
|-----------------|-----------------------------------|
| <i>__lo</i>     | Pointer to start of range.        |
| <i>__hi</i>     | Pointer to end of range.          |
| <i>__dfault</i> | Char to use if conversion fails.  |
| <i>__to</i>     | Pointer to the destination array. |

**Returns***\_\_hi*.

Definition at line 956 of file locale\_facets.h.

References std::ctype&lt; \_CharT &gt;::do\_narrow().

**4.678.3.14** `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const` [inline], [inherited]

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file ctype\_inline.h.

**4.678.3.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const` [inline], [inherited]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [`__lo`,`__hi`) for which is(m,char) is false.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file ctype\_inline.h.

**4.678.3.16** `const mask* std::ctype< char >::table ( ) const throw ()` [inline], [inherited]

Returns a pointer to the mask table provided to the constructor, or the default from classic\_table() if none was provided.

Definition at line 974 of file locale\_facets.h.

**4.678.3.17** `char_type std::ctype< char >::tolower ( char_type __c ) const` [inline], [inherited]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_c). do\_tolower() must always return the same result for the same input.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The lowercase char if convertible, else `__c`.

Definition at line 828 of file locale\_facets.h.

References `std::ctype<_CharT>::do_tolower()`.

**4.678.3.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo, const char_type * __hi ) const`  
`[inline],[inherited]`

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

`__hi`.

Definition at line 845 of file locale\_facets.h.

References `std::ctype<_CharT>::do_tolower()`.

**4.678.3.19** `char_type std::ctype< char >::toupper ( char_type __c ) const` `[inline],[inherited]`

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The uppercase char if convertible, else `__c`.

Definition at line 795 of file locale\_facets.h.

References `std::ctype<_CharT>::do_toupper()`.

**4.678.3.20** `const char_type* std::ctype< char >::toupper ( char_type * __lo, const char_type * __hi ) const`  
`[inline],[inherited]`

Convert array to uppercase.

This function converts each char in the range [\_\_lo,\_\_hi) to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(__lo, __hi)`. `do_toupper()` must always return the same result for the same input.

## Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

## Returns

`__hi`.

Definition at line 812 of file locale\_facets.h.

References `std::ctype< _CharT >::do_toupper()`.

**4.678.3.21** `char_type std::ctype< char >::widen ( char __c ) const` `[inline],[inherited]`

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

## Returns

The converted character.

Definition at line 865 of file locale\_facets.h.

References `std::ctype< _CharT >::do_widen()`.

**4.678.3.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const` `[inline],[inherited]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to first char in range.   |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi.`Definition at line 892 of file `locale_facets.h`.References `std::ctype<_CharT>::do_widen()`.

## 4.678.4 Member Data Documentation

4.678.4.1 `locale::id std::ctype<char>::id` `[static]`, `[inherited]`The facet id for `ctype<char>`Definition at line 696 of file `locale_facets.h`.4.678.4.2 `const size_t std::ctype<char>::table_size` `[static]`, `[inherited]`The size of the mask table. It is `SCHAR_MAX + 1`.Definition at line 698 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

4.679 `std::decay<_Tp>` Class Template Reference

## Public Types

- `typedef __decay_selector<__remove_type>::__type type`

## 4.679.1 Detailed Description

`template<typename _Tp>class std::decay<_Tp>``decay`Definition at line 1725 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

4.680 `std::decimal::decimal128` Class Reference

## Public Types

- `typedef float __decfloat128 __attribute__((mode(TD)))`

## Public Member Functions

- `decimal128 (decimal32 d32)`

- **decimal128** ([decimal64](#) d64)
- **decimal128** (float \_\_r)
- **decimal128** (double \_\_r)
- **decimal128** (long double \_\_r)
- **decimal128** (int \_\_z)
- **decimal128** (unsigned int \_\_z)
- **decimal128** (long \_\_z)
- **decimal128** (unsigned long \_\_z)
- **decimal128** (long long \_\_z)
- **decimal128** (unsigned long long \_\_z)
- [decimal128](#) (\_\_decfloat128 \_\_z)
- [\\_\\_decfloat128](#) \_\_getval (void)
- void \_\_setval (\_\_decfloat128 \_\_x)
- [decimal128](#) & **operator**\*= ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator**\*= ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator**\*= ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator**\*= (int \_\_rhs)
- [decimal128](#) & **operator**\*= (unsigned int \_\_rhs)
- [decimal128](#) & **operator**\*= (long \_\_rhs)
- [decimal128](#) & **operator**\*= (unsigned long \_\_rhs)
- [decimal128](#) & **operator**\*= (unsigned long long \_\_rhs)
- [decimal128](#) & **operator**\*= (long long \_\_rhs)
- [decimal128](#) & **operator**++ ()
- [decimal128](#) **operator**++ (int)
- [decimal128](#) & **operator**+= (int \_\_rhs)
- [decimal128](#) & **operator**+= ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator**+= ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator**+= ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator**+= (unsigned int \_\_rhs)
- [decimal128](#) & **operator**+= (long \_\_rhs)
- [decimal128](#) & **operator**+= (unsigned long \_\_rhs)
- [decimal128](#) & **operator**+= (long long \_\_rhs)
- [decimal128](#) & **operator**+= (unsigned long long \_\_rhs)
- [decimal128](#) & **operator**-- ()
- [decimal128](#) **operator**-- (int)
- [decimal128](#) & **operator**-= (long \_\_rhs)
- [decimal128](#) & **operator**-= (int \_\_rhs)
- [decimal128](#) & **operator**-= (long long \_\_rhs)
- [decimal128](#) & **operator**-= (unsigned long long \_\_rhs)
- [decimal128](#) & **operator**-= (unsigned long \_\_rhs)
- [decimal128](#) & **operator**-= ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator**-= (unsigned int \_\_rhs)
- [decimal128](#) & **operator**-= ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator**-= ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator**/= ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator**/= (long long \_\_rhs)
- [decimal128](#) & **operator**/= (unsigned long long \_\_rhs)
- [decimal128](#) & **operator**/= (unsigned long \_\_rhs)
- [decimal128](#) & **operator**/= ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator**/= (unsigned int \_\_rhs)
- [decimal128](#) & **operator**/= ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator**/= (long \_\_rhs)
- [decimal128](#) & **operator**/= (int \_\_rhs)

## 4.680.1 Detailed Description

## 3.2.4 Class decimal128.

Definition at line 393 of file decimal.

## 4.680.2 Constructor &amp; Destructor Documentation

## 4.680.2.1 std::decimal::decimal128::decimal128 ( \_\_decfloat128 \_\_z ) [inline]

Conforming extension: Conversion from scalar decimal type.

Definition at line 418 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 4.681 std::decimal::decimal32 Class Reference

## Public Types

- typedef float \_\_decfloat32 **\_\_attribute\_\_** ((mode(SD)))

## Public Member Functions

- **decimal32** ([decimal64](#) \_\_d64)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** (float \_\_r)
- **decimal32** (double \_\_r)
- **decimal32** (long double \_\_r)
- **decimal32** (int \_\_z)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (long long \_\_z)
- **decimal32** (unsigned long long \_\_z)
- [decimal32](#) (\_\_decfloat32 \_\_z)
- \_\_decfloat32 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat32 \_\_x)
- [decimal32](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator\*=** (int \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator\*=** (long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator\*=** (long long \_\_rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** (int \_\_rhs)

- [decimal32](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator+=** (long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator+=** (long long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** (long \_\_rhs)
- [decimal32](#) & **operator-=** (int \_\_rhs)
- [decimal32](#) & **operator-=** (long long \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator/=** (long long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator/=** (long \_\_rhs)
- [decimal32](#) & **operator/=** (int \_\_rhs)

#### 4.681.1 Detailed Description

##### 3.2.2 Class decimal32.

Definition at line 227 of file decimal.

#### 4.681.2 Constructor & Destructor Documentation

##### 4.681.2.1 std::decimal::decimal32::decimal32 ( \_\_decfloat32 \_\_z ) [inline]

Conforming extension: Conversion from scalar decimal type.

Definition at line 251 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 4.682 std::decimal::decimal64 Class Reference

### Public Types

- typedef float \_\_decfloat64 \_\_attribute\_\_((mode(DD)))

## Public Member Functions

- **decimal64** ([decimal32](#) d32)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (float \_\_r)
- **decimal64** (double \_\_r)
- **decimal64** (long double \_\_r)
- **decimal64** (int \_\_z)
- **decimal64** (unsigned int \_\_z)
- **decimal64** (long \_\_z)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (long long \_\_z)
- **decimal64** (unsigned long long \_\_z)
- [decimal64](#) (\_\_decfloat64 \_\_z)
- \_\_decfloat64 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- [decimal64](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator\*=** (int \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator\*=** (long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator\*=** (long long \_\_rhs)
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=** (int \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator+=** (long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator+=** (long long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=** (long \_\_rhs)
- [decimal64](#) & **operator-=** (int \_\_rhs)
- [decimal64](#) & **operator-=** (long long \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator/=** (long long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long \_\_rhs)

- [decimal64](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator/=** (long \_\_rhs)
- [decimal64](#) & **operator/=** (int \_\_rhs)

#### 4.682.1 Detailed Description

##### 3.2.3 Class decimal64.

Definition at line 310 of file decimal.

#### 4.682.2 Constructor & Destructor Documentation

##### 4.682.2.1 `std::decimal::decimal64::decimal64 ( __decfloat64 __z ) [inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 334 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 4.683 `std::default_delete<_Tp>` Struct Template Reference

### Public Member Functions

- `template<typename _Up, typename = typename enable_if<is_convertible<_Up*, _Tp*>::value>::type> default_delete (const default\_delete<_Up> &) noexcept`
- `void operator() (_Tp * __ptr) const`

#### 4.683.1 Detailed Description

`template<typename _Tp>struct std::default_delete<_Tp>`

Primary template, default\_delete.

Definition at line 54 of file unique\_ptr.h.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 4.684 `std::default_delete<_Tp[]>` Struct Template Reference

### Public Member Functions

- `template<typename _Up, typename = typename enable_if<!__is_derived_Tp<_Up>::value>::type> default_delete (const default\_delete<_Up[]> &) noexcept`
- `void operator() (_Tp * __ptr) const`

- `template<typename _Up >  
enable_if< __is_derived_Tp  
< _Up >::value >::type operator() (_Up *) const =delete`

#### 4.684.1 Detailed Description

`template<typename _Tp>struct std::default_delete< _Tp[]>`

Specialization, `default_delete`.

Definition at line 75 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 4.685 `std::defer_lock_t` Struct Reference

### 4.685.1 Detailed Description

Do not acquire ownership of the mutex.

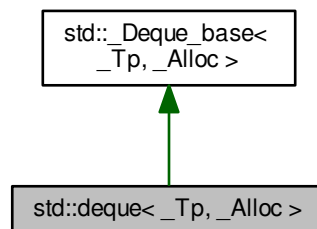
Definition at line 389 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 4.686 `std::deque< _Tp, _Alloc >` Class Template Reference

Inheritance diagram for `std::deque< _Tp, _Alloc >`:



### Public Types

- `typedef _Alloc allocator_type`
- `typedef _Base::const_iterator const_iterator`

- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator`  
    < `const_iterator` > **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator`  
    < `iterator` > **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- `deque()`
- `deque(const allocator_type &__a)`
- `deque(size_type __n)`
- `deque(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `deque(const deque &__x)`
- `deque(deque &&__x)`
- `deque(initializer_list<value_type> __l, const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
    `deque(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `~deque()` noexcept
- `void assign(size_type __n, const value_type &__val)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
    `void assign(_InputIterator __first, _InputIterator __last)`
- `void assign(initializer_list<value_type> __l)`
- `reference at(size_type __n)`
- `const_reference at(size_type __n) const`
- `reference back()`
- `const_reference back() const`
- `iterator begin()` noexcept
- `const_iterator begin() const` noexcept
- `const_iterator cbegin() const` noexcept
- `const_iterator cend() const` noexcept
- `void clear()` noexcept
- `const_reverse_iterator crbegin() const` noexcept
- `const_reverse_iterator crend() const` noexcept
- `template<typename... _Args>`  
    `iterator emplace(iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
    `void emplace_back(_Args &&... __args)`
- `template<typename... _Args>`  
    `void emplace_front(_Args &&... __args)`
- `bool empty() const` noexcept
- `iterator end()` noexcept
- `const_iterator end() const` noexcept

- iterator erase (iterator \_\_position)
- iterator erase (iterator \_\_first, iterator \_\_last)
- reference front ()
- const\_reference front () const
- allocator\_type get\_allocator () const noexcept
- iterator insert (iterator \_\_position, const value\_type &\_\_x)
- iterator insert (iterator \_\_position, value\_type &&\_\_x)
- void insert (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- void insert (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void insert (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type max\_size () const noexcept
- deque & operator= (const deque &\_\_x)
- deque & operator= (deque &&\_\_x)
- deque & operator= (initializer\_list< value\_type > \_\_l)
- reference operator[] (size\_type \_\_n)
- const\_reference operator[] (size\_type \_\_n) const
- void pop\_back ()
- void pop\_front ()
- void push\_back (const value\_type &\_\_x)
- void push\_back (value\_type &&\_\_x)
- void push\_front (const value\_type &\_\_x)
- void push\_front (value\_type &&\_\_x)
- reverse\_iterator rbegin () noexcept
- const\_reverse\_iterator rbegin () const noexcept
- reverse\_iterator rend () noexcept
- const\_reverse\_iterator rend () const noexcept
- void resize (size\_type \_\_new\_size)
- void resize (size\_type \_\_new\_size, const value\_type &\_\_x)
- void shrink\_to\_fit ()
- size\_type size () const noexcept
- void swap (deque &\_\_x)

#### Protected Types

- enum { **\_S\_initial\_map\_size** }
- typedef \_Alloc::template  
rebind<\_Tp \* >::other **\_Map\_alloc\_type**
- typedef pointer \* **\_Map\_pointer**

#### Protected Member Functions

- \_Tp \*\* **\_M\_allocate\_map** (size\_t \_\_n)
- \_Tp \* **\_M\_allocate\_node** ()
- template<typename \_InputIterator >  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, std::input\_iterator\_tag)
- template<typename \_ForwardIterator >  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, std::forward\_iterator\_tag)
- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)

- `template<typename _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Tp ** __nstart, _Tp ** __nfinish)`
- `void _M_deallocate_map (_Tp ** __p, size_t __n)`
- `void _M_deallocate_node (_Tp * __p)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`  
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator<_Tp> &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp ** __nstart, _Tp ** __nfinish)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type & __val)`
- `void _M_fill_initialize (const value_type & __value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type & __x)`
- `_Map_alloc_type _M_get_map_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_initialize_map (size_t)`
- `template<typename... _Args>`  
`iterator _M_insert_aux (iterator __pos, _Args &&... __args)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type & __x)`
- `template<typename _ForwardIterator >`  
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `template<typename _InputIterator >`  
`void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `bool _M_shrink_to_fit ()`
  
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
  
- `template<typename... _Args>`  
`void _M_push_back_aux (_Args &&... __args)`
- `template<typename... _Args>`  
`void _M_push_front_aux (_Args &&... __args)`

- void [\\_M\\_pop\\_back\\_aux](#) ()
- void [\\_M\\_pop\\_front\\_aux](#) ()
- [iterator \\_M\\_reserve\\_elements\\_at\\_front](#) (size\_type \_\_n)
- [iterator \\_M\\_reserve\\_elements\\_at\\_back](#) (size\_type \_\_n)
- void [\\_M\\_new\\_elements\\_at\\_front](#) (size\_type \_\_new\_elements)
- void [\\_M\\_new\\_elements\\_at\\_back](#) (size\_type \_\_new\_elements)
- void [\\_M\\_reserve\\_map\\_at\\_back](#) (size\_type \_\_nodes\_to\_add=1)
- void [\\_M\\_reserve\\_map\\_at\\_front](#) (size\_type \_\_nodes\_to\_add=1)
- void [\\_M\\_reallocate\\_map](#) (size\_type \_\_nodes\_to\_add, bool \_\_add\_at\_front)

#### Static Protected Member Functions

- static size\_t [\\_S\\_buffer\\_size](#) ()

#### Protected Attributes

- [\\_Deque\\_impl](#) [\\_M\\_impl](#)

#### 4.686.1 Detailed Description

template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> class std::deque< \_Tp, \_Alloc >

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

#### Template Parameters

|                        |                                                                 |
|------------------------|-----------------------------------------------------------------|
| <a href="#">_Tp</a>    | Type of element.                                                |
| <a href="#">_Alloc</a> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

`map_size` is at least 8. `map` is an array of `map_size` pointers-to-*nodes*. (The name `map` has nothing to do with the `std::map` class, and **nodes** should not be confused with `std::list`'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-`Tp`. If `Tp` is very large, there will be one `Tp` element per node (i.e., an *array* of one). For non-huge `Tp`'s, node size is inversely related to `Tp` size: the larger the `Tp`, the fewer `Tp`'s will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different `Tp`'s, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator *i*:
  - *i*.node points to a member of the map array. (Yes, you read that correctly: *i*.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - *i*.first == *\*(i.node)* (This points to the node (first *Tp* element).)
  - *i*.last == *i*.first + node\_size
  - *i*.cur is a pointer in the range [*i*.first, *i*.last). NOTE: the implication of this is that *i*.cur is always a dereferenceable pointer, even if *i* is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map\_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map\_size).
- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, \_Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 730 of file stl\_deque.h.

#### 4.686.2 Constructor & Destructor Documentation

4.686.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( )`  
`[inline]`

Default constructor creates no elements.

Definition at line 782 of file stl\_deque.h.

4.686.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( const allocator_type & __a )` `[inline], [explicit]`

Creates a deque with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

Definition at line 790 of file stl\_deque.h.

**4.686.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque ( size_type __n ) [inline],[explicit]`

Creates a deque with default constructed elements.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
|------------------|---------------------------------------------|

This constructor fills the deque with *n* default constructed elements.

Definition at line 802 of file stl\_deque.h.

**4.686.2.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque ( size_type __n, const value_type & __value, const allocator_type & __a = allocator_type() ) [inline]`

Creates a deque with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the deque with `__n` copies of `__value`.

Definition at line 814 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::_M_fill_initialize()`.

**4.686.2.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque ( const deque< _Tp, _Alloc > & __x ) [inline]`

Deque copy constructor.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The newly-created deque uses a copy of the allocation object used by `__x`.

Definition at line 841 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::begin()`, and `std::deque< _Tp, _Alloc >::end()`.

**4.686.2.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque ( deque< _Tp, _Alloc > && __x ) [inline]`

Deque move constructor.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The newly-created deque contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified deque.

Definition at line 855 of file stl\_deque.h.

4.686.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque ( initializer_list< value_type > __l, const allocator_type & __a = allocator_type() ) [inline]`

Builds a deque from an initializer list.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
| <code>__a</code> | An allocator object. |

Create a deque consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 869 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_range_initialize()`.

4.686.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> std::deque< _Tp, _Alloc >::deque ( _InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type() ) [inline]`

Builds a deque from a range.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__a</code>     | An allocator object. |

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 896 of file `std_deque.h`.

4.686.2.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::~~deque ( ) [inline], [noexcept]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 917 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`, and `std::deque< _Tp, _Alloc >::end()`.

### 4.686.3 Member Function Documentation

4.686.3.1 `template<typename _Tp, typename _Alloc > void deque::_M_fill_initialize ( const value_type & __value ) [protected]`

Fills the deque with copies of value.

#### Parameters

|                      |                |
|----------------------|----------------|
| <code>__value</code> | Initial value. |
|----------------------|----------------|

**Returns**

Nothing.

**Precondition**

\_M\_start and \_M\_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 351 of file deque.tcc.

References std::\_Destroy().

Referenced by std::deque< \_Tp, \_Alloc >::deque().

**4.686.3.2** `template<typename _Tp, typename _Alloc> void std:: Deque_base< _Tp, _Alloc >::M_initialize_map ( size_t __num_elements ) [protected], [inherited]`

Layout storage.

**Parameters**

|                             |                                                        |
|-----------------------------|--------------------------------------------------------|
| <code>__num_elements</code> | The count of T's for which to allocate space at first. |
|-----------------------------|--------------------------------------------------------|

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 582 of file stl\_deque.h.

References std::max().

**4.686.3.3** `template<typename _Tp, typename _Alloc> void deque::M_new_elements_at_back ( size_type __new_elements ) [protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 845 of file deque.tcc.

References std::size().

Referenced by std::deque< \_Tp, \_Alloc >::M\_reserve\_elements\_at\_back().

**4.686.3.4** `template<typename _Tp, typename _Alloc> void deque::M_new_elements_at_front ( size_type __new_elements ) [protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 820 of file deque.tcc.

References std::size().

Referenced by std::deque< \_Tp, \_Alloc >::M\_reserve\_elements\_at\_front().

**4.686.3.5** `template<typename _Tp, typename _Alloc> void deque::M_pop_back_aux ( ) [protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 501 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_back().

4.686.3.6 `template<typename _Tp, typename _Alloc > void deque::_M_pop_front_aux ( ) [protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 516 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_front().

4.686.3.7 `template<typename _Tp, typename _Alloc > template<typename... _Args> void deque::_M_push_back_aux ( _Args &&... __args ) [protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 435 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::push\_back().

4.686.3.8 `template<typename _Tp, typename _Alloc > template<typename... _Args> void deque::_M_push_front_aux ( _Args &&... __args ) [protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 469 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::push\_front().

4.686.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_range_check ( size_type __n ) const [inline], [protected]`

Safety check used only from at().

Definition at line 1265 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::size().

Referenced by std::deque< \_Tp, \_Alloc >::at().

4.686.3.10 `template<typename _Tp, typename _Alloc > template<typename _InputIterator > void deque::_M_range_initialize ( _InputIterator __first, _InputIterator __last, std::input_iterator_tag ) [protected]`

Fills the deque with whatever is in [first,last).

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

#### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push\_back on each value from the iterator.

Definition at line 377 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::deque().

4.686.3.11 `template<typename _Tp, typename _Alloc > template<typename _ForwardIterator > void deque::_M_range_initialize ( _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag ) [protected]`

Fills the deque with whatever is in [first,last).

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

#### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 397 of file `deque.tcc`.

References `std::_Destroy()`, `std::advance()`, and `std::distance()`.

4.686.3.12 `template<typename _Tp, typename _Alloc > void deque::_M_reallocate_map ( size_type __nodes_to_add, bool __add_at_front ) [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 870 of file `deque.tcc`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_map_at_back()`, and `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`.

4.686.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back ( size_type __n ) [inline], [protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1898 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

4.686.3.14 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front ( size_type __n ) [inline], [protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1888 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

4.686.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back ( size_type __nodes_to_add = 1 ) [inline], [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1924 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::\_M\_reallocate\_map().

**4.686.3.16** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::deque< \_Tp, \_Alloc >::\_M\_reserve\_map\_at\_front ( size\_type \_\_nodes\_to\_add = 1 ) [inline], [protected]

Memory-handling helpers for the major map.

Makes sure the \_M\_map has space for new nodes. Does not actually add the nodes. Can invalidate \_M\_map pointers. (And consequently, deque iterators.)

Definition at line 1932 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::\_M\_reallocate\_map().

**4.686.3.17** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::deque< \_Tp, \_Alloc >::assign ( size\_type \_\_n, const value\_type & \_\_val ) [inline]

Assigns a given value to a deque.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 978 of file stl\_deque.h.

Referenced by std::deque< \_Tp, \_Alloc >::operator=().

**4.686.3.18** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>> void std::deque< \_Tp, \_Alloc >::assign ( \_InputIterator \_\_first, \_InputIterator \_\_last ) [inline]

Assigns a range to a deque.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a deque with copies of the elements in the range [`__first`, `__last`).

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 997 of file stl\_deque.h.

**4.686.3.19** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> void std::deque< \_Tp, \_Alloc >::assign ( initializer\_list< value\_type > \_\_l ) [inline]

Assigns an initializer list to a deque.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills a deque with copies of the elements in the initializer\_list \_\_l.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 1022 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::assign().

Referenced by std::deque< \_Tp, \_Alloc >::assign().

**4.686.3.20** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> reference std::deque< \_Tp, \_Alloc >::at (size\_type \_\_n) [inline]

Provides access to the data contained in the deque.

#### Parameters

|     |                                                             |
|-----|-------------------------------------------------------------|
| __n | The index of the element for which data should be accessed. |
|-----|-------------------------------------------------------------|

#### Returns

Read/write reference to data.

#### Exceptions

|                   |                             |
|-------------------|-----------------------------|
| std::out_of_range | If __n is an invalid index. |
|-------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out\_of\_range if the check fails.

Definition at line 1284 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::\_M\_range\_check().

**4.686.3.21** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> const\_reference std::deque< \_Tp, \_Alloc >::at (size\_type \_\_n) const [inline]

Provides access to the data contained in the deque.

#### Parameters

|     |                                                             |
|-----|-------------------------------------------------------------|
| __n | The index of the element for which data should be accessed. |
|-----|-------------------------------------------------------------|

#### Returns

Read-only (constant) reference to data.

#### Exceptions

|                   |                             |
|-------------------|-----------------------------|
| std::out_of_range | If __n is an invalid index. |
|-------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out\_of\_range if the check fails.

Definition at line 1302 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::\_M\_range\_check().

**4.686.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::back ( )`  
`[inline]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1329 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::end()`.

**4.686.3.23** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::back ( ) const` `[inline]`

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1341 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::end()`.

**4.686.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::begin ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1037 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::deque()`, `std::deque< _Tp, _Alloc >::front()`, `std::deque< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::deque< _Tp, _Alloc >::~~deque()`.

**4.686.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::begin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1045 of file `stl_deque.h`.

**4.686.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::cbegin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1108 of file `stl_deque.h`.

**4.686.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::cend ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1117 of file `stl_deque.h`.

**4.686.3.28** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::clear ( )`  
`[inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1616 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

Referenced by std::deque< \_Tp, \_Alloc >::operator=().

**4.686.3.29** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1126 of file stl\_deque.h.

**4.686.3.30** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1135 of file stl\_deque.h.

**4.686.3.31** `template<typename _Tp, typename _Alloc > template<typename... _Args> deque< _Tp, _Alloc >::iterator deque::emplace ( iterator __position, _Args &&... __args )`

Inserts an object in deque before specified iterator.

#### Parameters

|                         |                             |
|-------------------------|-----------------------------|
| <code>__position</code> | An iterator into the deque. |
| <code>__args</code>     | Arguments.                  |

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location.

Definition at line 172 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::insert().

**4.686.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::deque< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]`

Returns true if the deque is empty. (Thus begin() would equal end().)

Definition at line 1228 of file stl\_deque.h.

**4.686.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::end ( ) [inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1054 of file stl\_deque.h.

Referenced by std::deque< \_Tp, \_Alloc >::back(), std::deque< \_Tp, \_Alloc >::deque(), std::deque< \_Tp, \_Alloc >::operator=(), std::operator==( ), and std::deque< \_Tp, \_Alloc >::~~deque().

4.686.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1063 of file `stl_deque.h`.

4.686.3.35 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase ( iterator __position )`

Remove element at given position.

#### Parameters

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 194 of file `deque.tcc`.

References `std::begin()`, `std::end()`, and `std::size()`.

4.686.3.36 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase ( iterator __first, iterator __last )`

Remove a range of elements.

#### Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

#### Returns

An iterator pointing to the element pointed to by `last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 217 of file `deque.tcc`.

References `std::begin()`, `std::end()`, and `std::size()`.

4.686.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1313 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

4.686.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1321 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

4.686.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::deque< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]`

Get a copy of the memory allocation object.

Definition at line 1028 of file `stl_deque.h`.

4.686.3.40 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::insert ( iterator __position, const value_type & __x )`

Inserts given value into deque before specified iterator.

#### Parameters

|                         |                             |
|-------------------------|-----------------------------|
| <code>__position</code> | An iterator into the deque. |
| <code>__x</code>        | Data to be inserted.        |

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 149 of file `deque.tcc`.

Referenced by `std::deque< _Tp, _Alloc >::resize()`.

4.686.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into deque before specified iterator.

#### Parameters

|                         |                             |
|-------------------------|-----------------------------|
| <code>__position</code> | An iterator into the deque. |
| <code>__x</code>        | Data to be inserted.        |

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1492 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::emplace()`.

4.686.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert ( iterator __p, initializer_list< value_type > __l ) [inline]`

Inserts an initializer list into the deque.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | An iterator into the deque. |
| <code>__l</code> | An initializer_list.        |

This function will insert copies of the data in the initializer\_list `__l` into the deque before the location specified by `__p`. This is known as *list insert*.

Definition at line 1505 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::insert()`.

Referenced by `std::deque< _Tp, _Alloc >::insert()`.

4.686.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert ( iterator __position, size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the deque.

## Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the deque.        |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

This function will insert a specified number of copies of the given data before the location specified by `__position`.

Definition at line 1519 of file `stl_deque.h`.

4.686.3.44 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> void std::deque< _Tp, _Alloc >::insert ( iterator __position, _InputIterator __first, _InputIterator __last ) [inline]`

Inserts a range into the deque.

## Parameters

|                         |                             |
|-------------------------|-----------------------------|
| <code>__position</code> | An iterator into the deque. |
| <code>__first</code>    | An input iterator.          |
| <code>__last</code>     | An input iterator.          |

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

Definition at line 1536 of file `stl_deque.h`.

4.686.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque< _Tp, _Alloc >::max_size ( ) const [inline],[noexcept]`

Returns the `size()` of the largest possible deque.

Definition at line 1147 of file `stl_deque.h`.

4.686.3.46 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc > & deque::operator= ( const deque< _Tp, _Alloc > & __x )`

Deque assignment operator.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

All the elements of `x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 93 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `std::deque< _Tp, _Alloc >::size()`, and `std::size()`.

**4.686.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= ( deque< _Tp, _Alloc > && __x ) [inline]`

Deque move assignment operator.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The contents of `__x` are moved into this deque (without copying). `__x` is a valid, but unspecified deque.

Definition at line 939 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::clear()`, and `std::deque< _Tp, _Alloc >::swap()`.

**4.686.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= ( initializer_list< value_type > __l ) [inline]`

Assigns an initializer list to a deque.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

This function fills a deque with copies of the elements in the `initializer_list __l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 960 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::assign()`.

**4.686.3.49** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::operator[] ( size_type __n ) [inline]`

Subscript access to the data contained in the deque.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

## Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1244 of file stl\_deque.h.

**4.686.3.50** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::operator[]( size_type __n ) const [inline]`

Subscript access to the data contained in the deque.

#### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1259 of file `stl_deque.h`.

**4.686.3.51** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::pop_back ( ) [inline]`

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1442 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_pop_back_aux()`.

**4.686.3.52** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1421 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_pop_front_aux()`.

**4.686.3.53** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::push_back ( const value_type & __x ) [inline]`

Add data to the end of the deque.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1390 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_push_back_aux()`.

**4.686.3.54** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::push_front ( const value_type & __x ) [inline]`

Add data to the front of the deque.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1359 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_push_front_aux()`.

**4.686.3.55** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1072 of file `stl_deque.h`.

**4.686.3.56** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1081 of file `stl_deque.h`.

**4.686.3.57** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1090 of file `stl_deque.h`.

**4.686.3.58** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1099 of file `stl_deque.h`.

**4.686.3.59** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::resize ( size_type __new_size ) [inline]`

Resizes the deque to the specified number of elements.

#### Parameters

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain. |
|-------------------------|----------------------------------------------|

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1161 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::size().

**4.686.3.60** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::resize ( size_type __new_size, const value_type & __x ) [inline]`

Resizes the deque to the specified number of elements.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain.      |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1183 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::insert(), and std::deque< \_Tp, \_Alloc >::size().

**4.686.3.61** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline]`

A non-binding request to reduce memory use.

Definition at line 1219 of file stl\_deque.h.

**4.686.3.62** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque< _Tp, _Alloc >::size ( ) const [inline], [noexcept]`

Returns the number of elements in the deque.

Definition at line 1142 of file stl\_deque.h.

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_range\_check(), std::deque< \_Tp, \_Alloc >::operator=(), std::operator==(), and std::deque< \_Tp, \_Alloc >::resize().

**4.686.3.63** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::swap ( deque< _Tp, _Alloc > & __x ) [inline]`

Swaps data with another deque.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A deque of the same element and allocator types. |
|------------------|--------------------------------------------------|

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(d1,d2) will feed to this function.

Definition at line 1596 of file stl\_deque.h.

Referenced by std::deque< \_Tp, \_Alloc >::operator=(), and std::swap().

The documentation for this class was generated from the following files:

- [stl\\_deque.h](#)
- [deque.tcc](#)

4.687 `std::discard_block_engine<_RandomNumberEngine, __p, __r>` Class Template Reference

## Public Types

- typedef  
    `_RandomNumberEngine::result_type` [result\\_type](#)

## Public Member Functions

- [discard\\_block\\_engine](#) ()
- [discard\\_block\\_engine](#) (const `_RandomNumberEngine` &\_\_rng)
- [discard\\_block\\_engine](#) (`_RandomNumberEngine` &&\_\_rng)
- [discard\\_block\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq`, typename = typename `std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`  
    [discard\\_block\\_engine](#) (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- void [seed](#) ()
- void [seed](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` >  
    void [seed](#) (`_Sseq` &\_\_q)

## Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

## Static Public Attributes

- static constexpr size\_t [block\\_size](#)
- static constexpr size\_t [used\\_block](#)

## Friends

- template<typename `_RandomNumberEngine1`, size\_t \_\_p1, size\_t \_\_r1, typename `_CharT`, typename `_Traits` >  
    [std::basic\\_ostream](#)< `_CharT`,  
    `_Traits` > & [operator<<](#) ([std::basic\\_ostream](#)< `_CharT`, `_Traits` > &\_\_os, const [std::discard\\_block\\_engine](#)< `_RandomNumberEngine1`, \_\_p1, \_\_r1 > &\_\_x)
- bool [operator==](#) (const [discard\\_block\\_engine](#) &\_\_lhs, const [discard\\_block\\_engine](#) &\_\_rhs)
- template<typename `_RandomNumberEngine1`, size\_t \_\_p1, size\_t \_\_r1, typename `_CharT`, typename `_Traits` >  
    [std::basic\\_istream](#)< `_CharT`,  
    `_Traits` > & [operator>>](#) ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &\_\_is, [std::discard\\_block\\_engine](#)< `_RandomNumberEngine1`, \_\_p1, \_\_r1 > &\_\_x)

## 4.687.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>class std::discard_block_engine< _RandomNumberEngine, __p, __r >
```

Produces random numbers from some base engine by discarding blocks of data.

0 <= \_\_r <= \_\_p

Definition at line 854 of file random.h.

## 4.687.2 Member Typedef Documentation

```
4.687.2.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type
std::discard_block_engine< _RandomNumberEngine, __p, __r >::result_type
```

The type of the generated random value.

Definition at line 857 of file random.h.

## 4.687.3 Constructor &amp; Destructor Documentation

```
4.687.3.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
_RandomNumberEngine, __p, __r >::discard_block_engine () [inline]
```

Constructs a default discard\_block\_engine engine.

The underlying engine is default constructed as well.

Definition at line 872 of file random.h.

```
4.687.3.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
_RandomNumberEngine, __p, __r >::discard_block_engine (const _RandomNumberEngine & __rng) [inline],
[explicit]
```

Copy constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

## Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 882 of file random.h.

```
4.687.3.3 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
_RandomNumberEngine, __p, __r >::discard_block_engine (_RandomNumberEngine && __rng) [inline],
[explicit]
```

Move constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

## Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 892 of file random.h.

```
4.687.3.4 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
 _RandomNumberEngine, __p, __r>::discard_block_engine(result_type __s) [inline],[explicit]
```

Seed constructs a discard\_block\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

#### Parameters

|     |                                         |
|-----|-----------------------------------------|
| __s | A seed value for the base class engine. |
|-----|-----------------------------------------|

Definition at line 902 of file random.h.

```
4.687.3.5 template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq, typename
 = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq,
 _RandomNumberEngine>::value> ::type> std::discard_block_engine< _RandomNumberEngine, __p, __r
 >::discard_block_engine(_Sseq & __q) [inline],[explicit]
```

Generator construct a discard\_block\_engine engine.

#### Parameters

|     |                  |
|-----|------------------|
| __q | A seed sequence. |
|-----|------------------|

Definition at line 915 of file random.h.

### 4.687.4 Member Function Documentation

```
4.687.4.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> const _RandomNumberEngine&
 std::discard_block_engine< _RandomNumberEngine, __p, __r>::base() const [inline],[noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 959 of file random.h.

```
4.687.4.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<
 _RandomNumberEngine, __p, __r>::discard(unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 980 of file random.h.

```
4.687.4.3 template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type
 std::discard_block_engine< _RandomNumberEngine, __p, __r>::max() [inline],[static]
```

Gets the maximum value in the generated random number range.

Definition at line 973 of file random.h.

References std::max().

```
4.687.4.4 template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type
 std::discard_block_engine< _RandomNumberEngine, __p, __r>::min() [inline],[static]
```

Gets the minimum value in the generated random number range.

Definition at line 966 of file random.h.

References std::min().

4.687.4.5 `template<typename _RandomNumberEngine, size_t __p, size_t __r> discard_block_engine<_RandomNumberEngine, __p, __r>::result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::operator() ( )`

Gets the next value in the generated random number sequence.

Definition at line 688 of file bits/random.tcc.

4.687.4.6 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed ( ) [inline]`

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 924 of file random.h.

4.687.4.7 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed ( result_type __s ) [inline]`

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 935 of file random.h.

4.687.4.8 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed ( _Sseq & __q ) [inline]`

Reseeds the discard\_block\_engine object with the given seed sequence.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__q</code> | A seed generator function. |
|------------------|----------------------------|

Definition at line 948 of file random.h.

#### 4.687.5 Friends And Related Function Documentation

4.687.5.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> & __x ) [friend]`

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A discard_block_engine random number generator engine. |

**Returns**

The output stream with the state of \_\_\_x inserted or in an error state.

```
4.687.5.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> bool operator==(const discard_block_engine<
 _RandomNumberEngine, __p, __r> & __lhs, const discard_block_engine< _RandomNumberEngine, __p, __r> & __rhs
) [friend]
```

Compares two discard\_block\_engine random number generator objects of the same type for equality.

**Parameters**

|       |                                                              |
|-------|--------------------------------------------------------------|
| __lhs | A discard_block_engine random number generator object.       |
| __rhs | Another discard_block_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1004 of file random.h.

```
4.687.5.3 template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1 ,
 size_t __p1, size_t __r1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits> & __is, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1
 > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_\_x from the input stream \_\_\_is.

**Parameters**

|      |                                                        |
|------|--------------------------------------------------------|
| __is | An input stream.                                       |
| __x  | A discard_block_engine random number generator engine. |

**Returns**

The input stream with the state of \_\_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**4.688 std::discrete\_distribution< \_IntType > Class Template Reference****Classes**

- struct [param\\_type](#)

**Public Types**

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- template<typename \_InputIterator >  
**discrete\_distribution** (\_InputIterator \_\_wbegin, \_InputIterator \_\_wend)
- **discrete\_distribution** (initializer\_list< double > \_\_wl)
- template<typename \_Func >  
**discrete\_distribution** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, \_Func \_\_fw)
- **discrete\_distribution** (const param\_type &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** (result\_type \* \_\_f, result\_type \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- result\_type **max** () const
- result\_type **min** () const
- template<typename \_UniformRandomNumberGenerator >  
result\_type **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
result\_type **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- param\_type **param** () const
- void **param** (const param\_type &\_\_param)
- std::vector< double > **probabilities** () const
- void **reset** ()

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & **operator<<** (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::discrete\_distribution< \_IntType1 > &\_\_x)
- bool **operator==** (const discrete\_distribution &\_\_d1, const discrete\_distribution &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
std::basic\_istream< \_CharT, \_Traits > & **operator>>** (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::discrete\_distribution< \_IntType1 > &\_\_x)

## 4.688.1 Detailed Description

template<typename \_IntType = int>class std::discrete\_distribution< \_IntType >

A discrete\_distribution random number distribution.

The formula for the discrete probability mass function is

Definition at line 5250 of file random.h.

## 4.688.2 Member Typedef Documentation

## 4.688.2.1 template&lt;typename \_IntType = int&gt; typedef \_IntType std::discrete\_distribution&lt; \_IntType &gt;::result\_type

The type of the range of the distribution.

Definition at line 5253 of file random.h.

## 4.688.3 Member Function Documentation

4.688.3.1 template<typename \_IntType = int> result\_type std::discrete\_distribution< \_IntType >::max ( ) const  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 5370 of file random.h.

References std::vector< \_Tp, \_Alloc >::empty(), and std::vector< \_Tp, \_Alloc >::size().

4.688.3.2 template<typename \_IntType = int> result\_type std::discrete\_distribution< \_IntType >::min ( ) const  
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 5363 of file random.h.

4.688.3.3 template<typename \_IntType = int> template<typename UniformRandomNumberGenerator > result\_type  
std::discrete\_distribution< \_IntType >::operator() ( UniformRandomNumberGenerator & \_urng ) [inline]

Generating functions.

Definition at line 5381 of file random.h.

References std::discrete\_distribution< \_IntType >::operator()().

Referenced by std::discrete\_distribution< \_IntType >::operator()().

4.688.3.4 template<typename \_IntType = int> param\_type std::discrete\_distribution< \_IntType >::param ( ) const  
[inline]

Returns the parameter set of the distribution.

Definition at line 5348 of file random.h.

4.688.3.5 template<typename \_IntType = int> void std::discrete\_distribution< \_IntType >::param ( const param\_type &  
\_param ) [inline]

Sets the parameter set of the distribution.

## Parameters

|               |                                            |
|---------------|--------------------------------------------|
| <u>_param</u> | The new parameter set of the distribution. |
|---------------|--------------------------------------------|

Definition at line 5356 of file random.h.

## 4.688.3.6 template&lt;typename \_IntType = int&gt; std::vector&lt;double&gt; std::discrete\_distribution&lt; \_IntType &gt;::probabilities ( ) const [inline]

Returns the probabilities of the distribution.

Definition at line 5338 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

4.688.3.7 `template<typename _IntType = int> void std::discrete_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 5331 of file random.h.

#### 4.688.4 Friends And Related Function Documentation

4.688.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::discrete_distribution< _IntType1 > & __x ) [friend]`

Inserts a `discrete_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>discrete_distribution</code> random number distribution. |

##### Returns

The output stream with the state of `__x` inserted or in an error state.

4.688.4.2 `template<typename _IntType = int> bool operator== ( const discrete_distribution< _IntType > & __d1, const discrete_distribution< _IntType > & __d2 ) [friend]`

Return true if two discrete distributions have the same parameters.

Definition at line 5416 of file random.h.

4.688.4.3 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::discrete_distribution< _IntType1 > & __x ) [friend]`

Extracts a `discrete_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                     |
| <code>__x</code>  | A <code>discrete_distribution</code> random number generator engine. |

##### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.689 std::discrete\_distribution< \_IntType >::param\_type Struct Reference

### Public Types

- typedef [discrete\\_distribution](#)  
< \_IntType > **distribution\_type**

### Public Member Functions

- template<typename \_InputIterator >  
**param\_type** (\_InputIterator \_\_wbegin, \_InputIterator \_\_wend)
- **param\_type** ([initializer\\_list](#)< double > \_\_wil)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default
- [std::vector](#)< double > **probabilities** () const

### Friends

- class **discrete\_distribution**< \_IntType >
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.689.1 Detailed Description

template<typename \_IntType = int>struct std::discrete\_distribution< \_IntType >::param\_type

Parameter type.

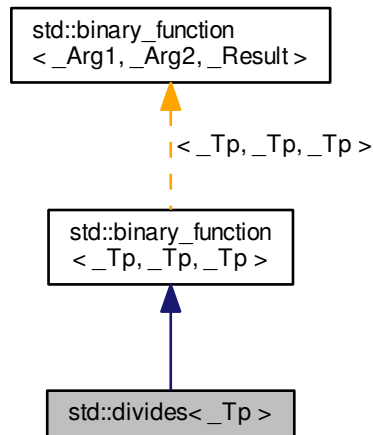
Definition at line 5259 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.690 std::divides&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::divides< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.690.1 Detailed Description

```
template<typename _Tp>struct std::divides< _Tp >
```

One of the [math functors](#).

Definition at line 167 of file `stl_function.h`.

## 4.690.2 Member Typedef Documentation

4.690.2.1 typedef `_Tp` `std::binary_function<_Tp, _Tp, _Tp>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.690.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.690.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

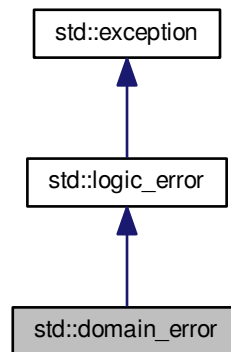
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.691 std::domain\_error Class Reference

Inheritance diagram for `std::domain_error`:



### Public Member Functions

- **domain\_error** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

### 4.691.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 74 of file `stdexcept`.

### 4.691.2 Member Function Documentation

4.691.2.1 `virtual const char* std::logic_error::what ( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

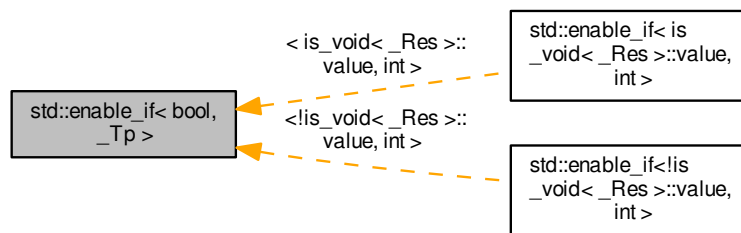
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 4.692 `std::enable_if< bool, _Tp >` Struct Template Reference

Inheritance diagram for `std::enable_if< bool, _Tp >`:



### 4.692.1 Detailed Description

```
template<bool, typename _Tp = void>struct std::enable_if< bool, _Tp >
```

Define a member typedef `type` only if a boolean constant is true.

Definition at line 1766 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.693 `std::enable_shared_from_this< _Tp >` Class Template Reference

### Public Member Functions

- [shared\\_ptr< \\_Tp >](#) **shared\_from\_this** ()
- [shared\\_ptr< const \\_Tp >](#) **shared\_from\_this** () const

### Protected Member Functions

- **enable\_shared\_from\_this** (const [enable\\_shared\\_from\\_this](#) &) noexcept
- [enable\\_shared\\_from\\_this](#) & **operator=** (const [enable\\_shared\\_from\\_this](#) &) noexcept

## Friends

- `template<typename _Tp1 >`  
`void __enable_shared_from_this_helper (const __shared_count<> &__pn, const enable\_shared\_from\_this *__pe, const _Tp1 *__px) noexcept`

## 4.693.1 Detailed Description

`template<typename _Tp>class std::enable_shared_from_this< _Tp >`

Base class allowing use of member function `shared_from_this`.

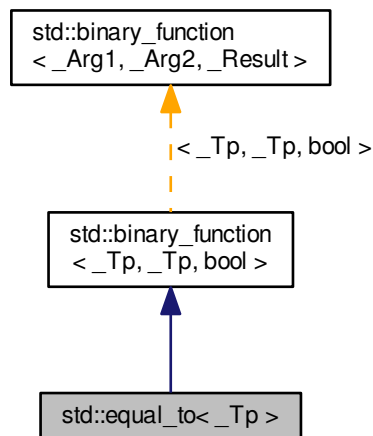
Definition at line 541 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 4.694 std::equal\_to&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for `std::equal_to< _Tp >`:



## Public Types

- `typedef _Tp first\_argument\_type`
- `typedef bool result\_type`
- `typedef _Tp second\_argument\_type`

## Public Member Functions

- `bool operator()` (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.694.1 Detailed Description

```
template<typename _Tp>struct std::equal_to< _Tp >
```

One of the [comparison functors](#).

Definition at line 204 of file `stl_function.h`.

## 4.694.2 Member Typedef Documentation

4.694.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.694.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.694.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.695 `std::error_category` Class Reference

## Public Member Functions

- `error_category` (const [error\\_category](#) &)=delete
- virtual [error\\_condition](#) `default_error_condition` (int \_\_i) const noexcept
- virtual bool `equivalent` (int \_\_i, const [error\\_condition](#) &\_\_cond) const noexcept
- virtual bool `equivalent` (const [error\\_code](#) &\_\_code, int \_\_i) const noexcept
- virtual [string](#) `message` (int) const =0
- virtual const char \* `name` () const noexcept=0
- bool `operator!=` (const [error\\_category](#) &\_\_other) const noexcept
- bool `operator<` (const [error\\_category](#) &\_\_other) const noexcept
- [error\\_category](#) & `operator=` (const [error\\_category](#) &)=delete
- bool `operator==` (const [error\\_category](#) &\_\_other) const noexcept

## 4.695.1 Detailed Description

`error_category`

Definition at line 66 of file `system_error`.

The documentation for this class was generated from the following file:

- [system\\_error](#)

4.696 `std::error_code` Struct Reference

## Public Member Functions

- **`error_code`** (int \_\_v, const [error\\_category](#) &\_\_cat) noexcept
- template<typename \_ErrorCodeEnum , typename = typename enable\_if<is\_error\_code\_enum<\_ErrorCodeEnum>::value>::type>  
**`error_code`** (\_ErrorCodeEnum \_\_e) noexcept
- void **`assign`** (int \_\_v, const [error\\_category](#) &\_\_cat) noexcept
- const [error\\_category](#) & **`category`** () const noexcept
- void **`clear`** () noexcept
- [error\\_condition](#) **`default_error_condition`** () const noexcept
- [string](#) **`message`** () const
- **`operator bool`** () const noexcept
- template<typename \_ErrorCodeEnum >  
[enable\\_if](#)< [is\\_error\\_code\\_enum](#)  
< \_ErrorCodeEnum >::value,  
[error\\_code](#) & >::type **`operator=`** (\_ErrorCodeEnum \_\_e) noexcept
- int **`value`** () const noexcept

## Friends

- class **`hash`**< **`error_code`** >

## 4.696.1 Detailed Description

`error_code`

Definition at line 116 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

4.697 `std::error_condition` Struct Reference

## Public Member Functions

- **`error_condition`** (int \_\_v, const [error\\_category](#) &\_\_cat) noexcept
- template<typename \_ErrorConditionEnum , typename = typename enable\_if<is\_error\_condition\_enum<\_ErrorConditionEnum>::value>::type>  
**`error_condition`** (\_ErrorConditionEnum \_\_e) noexcept
- void **`assign`** (int \_\_v, const [error\\_category](#) &\_\_cat) noexcept

- const [error\\_category](#) & **category** () const noexcept
- void **clear** () noexcept
- [string](#) **message** () const
- **operator bool** () const noexcept
- template<typename \_ErrorConditionEnum >  
  [enable\\_if](#)  
  < [is\\_error\\_condition\\_enum](#)  
  < \_ErrorConditionEnum >::value,  
  [error\\_condition](#) & >::type **operator=** (\_ErrorConditionEnum \_\_e) noexcept
- int **value** () const noexcept

#### 4.697.1 Detailed Description

`error_condition`

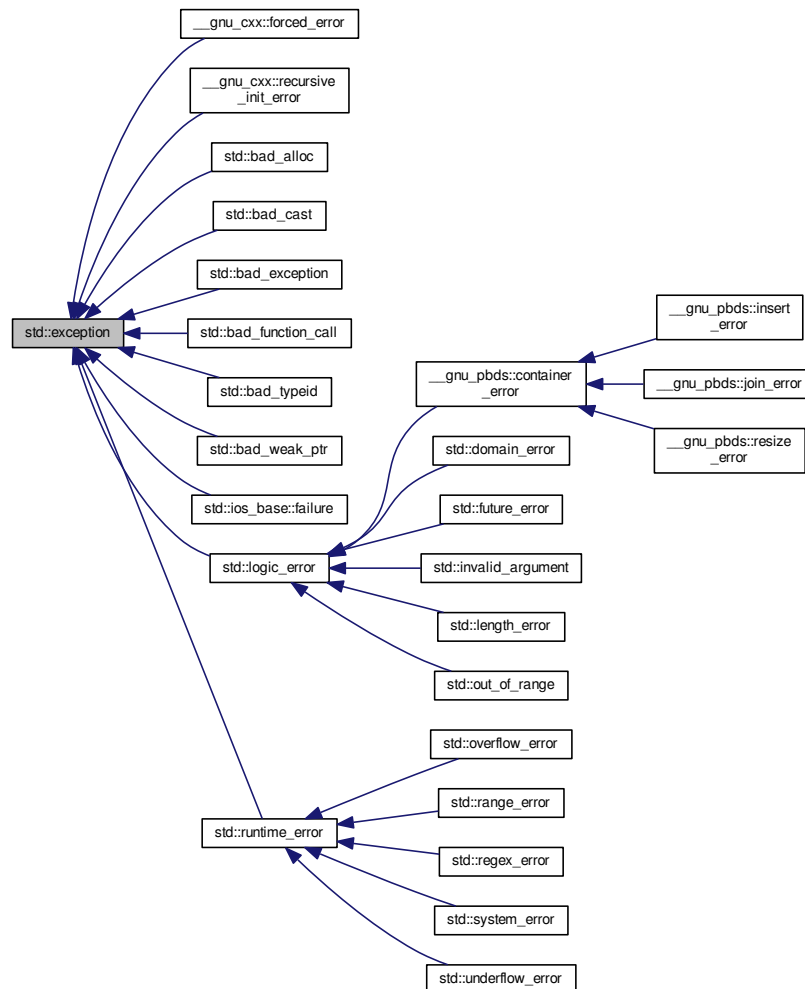
Definition at line 193 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 4.698 std::exception Class Reference

Inheritance diagram for std::exception:



## Public Member Functions

- virtual const char \* [what](#) () const noexcept

## 4.698.1 Detailed Description

Base class for all library exceptions.

This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 60 of file exception.

## 4.698.2 Member Function Documentation

4.698.2.1 `virtual const char* std::exception::what ( ) const` `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_function_call`, `std::ios_base::failure`, `std::bad_typeid`, `std::bad_cast`, `std::runtime_error`, `std::future_error`, `std::bad_exception`, `std::logic_error`, `std::bad_weak_ptr`, and `std::bad_alloc`.

The documentation for this class was generated from the following file:

- [exception](#)

4.699 `std::exponential_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [exponential\\_distribution](#) (const [result\\_type](#) &\_\_lambda=[result\\_type](#)(1))
- **`exponential_distribution`** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **`__generate`** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **`__generate`** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
void **`__generate`** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- `_RealType` [lambda](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool [operator==](#) (const [exponential\\_distribution](#) &\_\_d1, const [exponential\\_distribution](#) &\_\_d2)

## 4.699.1 Detailed Description

```
template<typename _RealType = double> class std::exponential_distribution< _RealType >
```

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

|                    |                         |
|--------------------|-------------------------|
| Mean               | $\frac{1}{\lambda}$     |
| Median             | $\frac{\ln 2}{\lambda}$ |
| Mode               | <i>zero</i>             |
| Range              | $[0, \infty]$           |
| Standard Deviation | $\frac{1}{\lambda}$     |

Table 1849: Distribution Statistics

Definition at line 4645 of file random.h.

## 4.699.2 Member Typedef Documentation

```
4.699.2.1 template<typename _RealType = double> typedef _RealType std::exponential_distribution< _RealType
>::result_type
```

The type of the range of the distribution.

Definition at line 4648 of file random.h.

## 4.699.3 Constructor &amp; Destructor Documentation

```
4.699.3.1 template<typename _RealType = double> std::exponential_distribution< _RealType
>::exponential_distribution (const result_type & __lambda = result_type(1)) [inline],
[explicit]
```

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

Definition at line 4683 of file random.h.

## 4.699.4 Member Function Documentation

```
4.699.4.1 template<typename _RealType = double> _RealType std::exponential_distribution< _RealType >::lambda ()
const [inline]
```

Returns the inverse scale parameter of the distribution.

Definition at line 4704 of file random.h.

```
4.699.4.2 template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::max ()
const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4733 of file random.h.

References std::numeric\_limits<\_Tp>::max().

4.699.4.3 `template<typename _RealType = double> result_type std::exponential_distribution<_RealType>::min ( )`  
`const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4726 of file random.h.

4.699.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type`  
`std::exponential_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng )`  
`[inline]`

Generating functions.

Definition at line 4741 of file random.h.

References std::exponential\_distribution<\_RealType>::operator()().

Referenced by std::exponential\_distribution<\_RealType>::operator()().

4.699.4.5 `template<typename _RealType = double> param_type std::exponential_distribution<_RealType>::param ( )`  
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 4711 of file random.h.

Referenced by std::operator>>().

4.699.4.6 `template<typename _RealType = double> void std::exponential_distribution<_RealType>::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4719 of file random.h.

4.699.4.7 `template<typename _RealType = double> void std::exponential_distribution<_RealType>::reset ( )`  
`[inline]`

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4698 of file random.h.

#### 4.699.5 Friends And Related Function Documentation

4.699.5.1 `template<typename _RealType = double> bool operator==( const exponential_distribution<_RealType> & __d1,`  
`const exponential_distribution<_RealType> & __d2 ) [friend]`

Return true if two exponential distributions have the same parameters.

Definition at line 4781 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.700 `std::exponential_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [exponential\\_distribution](#)  
`<_RealType> distribution_type`

### Public Member Functions

- **param\_type** (`_RealType __lambda=_RealType(1)`)
- `_RealType lambda` () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.700.1 Detailed Description

`template<typename _RealType = double> struct std::exponential_distribution<_RealType>::param_type`

Parameter type.

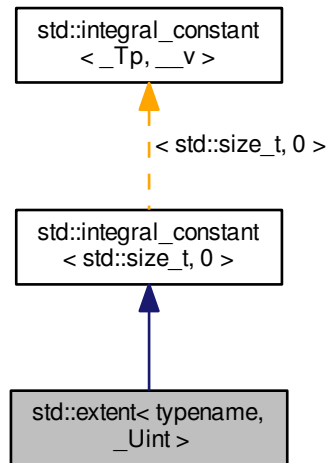
Definition at line 4654 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.701 `std::extent< typename, _UInt >` Struct Template Reference

Inheritance diagram for `std::extent< typename, _UInt >`:



## Public Types

- typedef [integral\\_constant](#)  
    `< std::size_t, __v > type`
- typedef `std::size_t value_type`

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `std::size_t value`

## 4.701.1 Detailed Description

`template<typename, unsigned _UInt> struct std::extent< typename, _UInt >`

`extent`

Definition at line 1256 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.702 std::extreme\_value\_distribution&lt; \_RealType &gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **extreme\_value\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [extreme\\_value\\_distribution](#) &\_\_d1, const [extreme\\_value\\_distribution](#) &\_\_d2)

## 4.702.1 Detailed Description

template<typename \_RealType = double>class std::extreme\_value\_distribution< \_RealType >

A extreme\_value\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b} \exp\left(\frac{a-x}{b}\right) - \exp\left(\frac{a-x}{b}\right)$$

Definition at line 5050 of file random.h.

## 4.702.2 Member Typedef Documentation

4.702.2.1 `template<typename _RealType = double> typedef _RealType std::extreme_value_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 5053 of file random.h.

## 4.702.3 Member Function Documentation

4.702.3.1 `template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::a ( ) const [inline]`

Return the  $a$  parameter of the distribution.

Definition at line 5108 of file random.h.

4.702.3.2 `template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::b ( ) const [inline]`

Return the  $b$  parameter of the distribution.

Definition at line 5115 of file random.h.

4.702.3.3 `template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5144 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

4.702.3.4 `template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5137 of file random.h.

References `std::numeric_limits<_Tp>::min()`.

4.702.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::extreme_value_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5152 of file random.h.

References `std::extreme_value_distribution<_RealType>::operator()()`.

Referenced by `std::extreme_value_distribution<_RealType>::operator()()`.

4.702.3.6 `template<typename _RealType = double> param_type std::extreme_value_distribution< _RealType >::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 5122 of file random.h.

Referenced by std::operator>>().

4.702.3.7 `template<typename _RealType = double> void std::extreme_value_distribution< _RealType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 5130 of file random.h.

4.702.3.8 `template<typename _RealType = double> void std::extreme_value_distribution< _RealType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 5101 of file random.h.

#### 4.702.4 Friends And Related Function Documentation

4.702.4.1 `template<typename _RealType = double> bool operator== ( const extreme_value_distribution< _RealType > & __d1, const extreme_value_distribution< _RealType > & __d2 ) [friend]`

Return true if two extreme value distributions have the same parameters.

Definition at line 5187 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 4.703 std::extreme\_value\_distribution< \_RealType >::param\_type Struct Reference

#### Public Types

- typedef [extreme\\_value\\_distribution](#) < \_RealType > **distribution\_type**

#### Public Member Functions

- **param\_type** ( \_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

#### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.703.1 Detailed Description

`template<typename _RealType = double> struct std::extreme_value_distribution<_RealType>::param_type`

Parameter type.

Definition at line 5059 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.704 `std::fisher_f_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **`fisher_f_distribution`** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- **`fisher_f_distribution`** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
void **`generate`** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
void **`generate`** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
void **`generate`** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, `_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
void **`generate`** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, `_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_RealType m () const`
- [result\\_type](#) **`max`** () const
- [result\\_type](#) **`min`** () const
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type](#) **`param`** () const
- void **`param`** (const [param\\_type](#) &\_\_param)
- void **`reset`** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
std::basic\_ostream<\_CharT,  
\_Traits> & operator<< (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const std::fisher\_f\_distribution<\_RealType1> &\_\_x)
- bool operator== (const fisher\_f\_distribution &\_\_d1, const fisher\_f\_distribution &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
std::basic\_istream<\_CharT,  
\_Traits> & operator>> (std::basic\_istream<\_CharT, \_Traits> &\_\_is, std::fisher\_f\_distribution<\_RealType1> &\_\_x)

## 4.704.1 Detailed Description

template<typename \_RealType = double>class std::fisher\_f\_distribution<\_RealType>

A fisher\_f\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m,n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 3130 of file random.h.

## 4.704.2 Member Typedef Documentation

4.704.2.1 template<typename \_RealType = double> typedef \_RealType std::fisher\_f\_distribution<\_RealType>::result\_type

The type of the range of the distribution.

Definition at line 3133 of file random.h.

## 4.704.3 Member Function Documentation

4.704.3.1 template<typename \_RealType = double> result\_type std::fisher\_f\_distribution<\_RealType>::max ( ) const  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 3224 of file random.h.

References std::numeric\_limits<\_Tp>::max().

4.704.3.2 template<typename \_RealType = double> result\_type std::fisher\_f\_distribution<\_RealType>::min ( ) const  
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3217 of file random.h.

4.704.3.3 template<typename \_RealType = double> template<typename UniformRandomNumberGenerator> result\_type  
std::fisher\_f\_distribution<\_RealType>::operator() ( UniformRandomNumberGenerator &\_\_urng ) [inline]

Generating functions.

Definition at line 3232 of file random.h.

4.704.3.4 `template<typename _RealType = double> param_type std::fisher_f_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 3202 of file random.h.

4.704.3.5 `template<typename _RealType = double> void std::fisher_f_distribution<_RealType>::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3210 of file random.h.

4.704.3.6 `template<typename _RealType = double> void std::fisher_f_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3181 of file random.h.

References `std::gamma_distribution<_RealType>::reset()`.

#### 4.704.4 Friends And Related Function Documentation

4.704.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( ( std::basic_ostream<_CharT, _Traits> & __os, const std::fisher_f_distribution<_RealType1> & __x ) [friend]`

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>fisher_f_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.704.4.2 `template<typename _RealType = double> bool operator==( const fisher_f_distribution<_RealType> & __d1, const fisher_f_distribution<_RealType> & __d2 ) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3280 of file random.h.

4.704.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::fisher_f_distribution<_RealType1> & __x ) [friend]`

Extracts a `fisher_f_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <code>__is</code> | An input stream.                                        |
| <code>__x</code>  | A fisher_f_distribution random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.705 std::fisher\_f\_distribution&lt;\_RealType&gt;::param\_type Struct Reference

## Public Types

- typedef [fisher\\_f\\_distribution](#)  
<\_RealType> **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_m=\_RealType(1), \_RealType \_\_n=\_RealType(1))
- \_RealType **m** () const
- \_RealType **n** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.705.1 Detailed Description

```
template<typename _RealType = double> struct std::fisher_f_distribution<_RealType>::param_type
```

Parameter type.

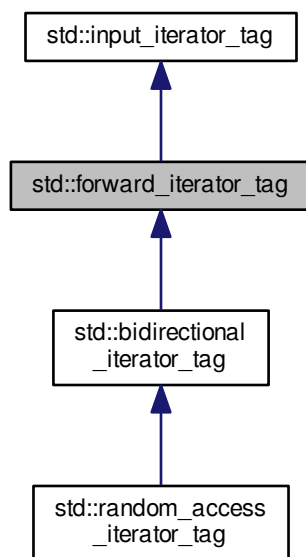
Definition at line 3139 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.706 std::forward\_iterator\_tag Struct Reference

Inheritance diagram for std::forward\_iterator\_tag:



### 4.706.1 Detailed Description

Forward iterators support a superset of input iterator operations.

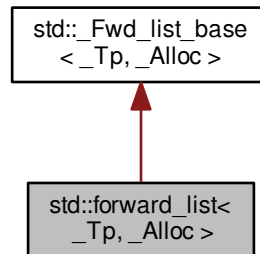
Definition at line 95 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.707 std::forward\_list&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::forward\_list< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Fwd_list_const_iterator< _Tp >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `forward_list` (`const _Alloc &__al= _Alloc()`)
- `forward_list` (`const forward_list &__list, const _Alloc &__al`)
- `forward_list` (`forward_list &&__list, const _Alloc &__al`) noexcept(`_Node_alloc_traits`)
- `forward_list` (`size_type __n, const _Alloc &__al= _Alloc()`)
- `forward_list` (`size_type __n, const _Tp &__value, const _Alloc &__al= _Alloc()`)
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`forward_list` (`_InputIterator __first, _InputIterator __last, const _Alloc &__al= _Alloc()`)
- `forward_list` (`const forward_list &__list`)
- `forward_list` (`forward_list &&__list`) noexcept
- `forward_list` (`std::initializer_list< _Tp > __il, const _Alloc &__al= _Alloc()`)
- `~forward_list` () noexcept
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`void assign` (`_InputIterator __first, _InputIterator __last`)

- void [assign](#) (size\_type \_\_n, const \_Tp &\_\_val)
- void [assign](#) (std::initializer\_list<\_Tp> \_\_il)
- [iterator before\\_begin](#) () noexcept
- [const\\_iterator before\\_begin](#) () const noexcept
- [iterator begin](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [const\\_iterator cbefore\\_begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- template<typename... \_Args>  
  [iterator emplace\\_after](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
  void [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [iterator erase\\_after](#) (const\_iterator \_\_pos)
- [iterator erase\\_after](#) (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- [iterator insert\\_after](#) (const\_iterator \_\_pos, const \_Tp &\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, \_Tp &&\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  [iterator insert\\_after](#) (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, std::initializer\_list<\_Tp> \_\_il)
- size\_type [max\\_size](#) () const noexcept
- void [merge](#) (forward\_list &&\_\_list)
- void [merge](#) (forward\_list &\_\_list)
- template<typename \_Comp>  
  void [merge](#) (forward\_list &&\_\_list, \_Comp \_\_comp)
- template<typename \_Comp>  
  void [merge](#) (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & [operator=](#) (const forward\_list &\_\_list)
- forward\_list & [operator=](#) (forward\_list &&\_\_list) noexcept(\_Node\_alloc\_traits
- forward\_list & [operator=](#) (std::initializer\_list<\_Tp> \_\_il)
- void [pop\\_front](#) ()
- void [push\\_front](#) (const \_Tp &\_\_val)
- void [push\\_front](#) (\_Tp &&\_\_val)
- void [remove](#) (const \_Tp &\_\_val)
- template<typename \_Pred>  
  void [remove\\_if](#) (\_Pred \_\_pred)
- void [resize](#) (size\_type \_\_sz)
- void [resize](#) (size\_type \_\_sz, const value\_type &\_\_val)
- void [reverse](#) () noexcept
- void [sort](#) ()
- template<typename \_Comp>  
  void [sort](#) (\_Comp \_\_comp)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&\_\_list)

- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&, const\_iterator \_\_before, const\_iterator \_\_last)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &, const\_iterator \_\_before, const\_iterator \_\_last)
- void **swap** (forward\_list &\_\_list) noexcept(\_Node\_alloc\_traits
- void **unique** ()
- template<typename \_BinPred >  
void **unique** (\_BinPred \_\_binary\_pred)

#### Private Member Functions

- template<typename... \_Args>  
\_Node \* **\_M\_create\_node** (\_Args &&... \_\_args)
- \_Fwd\_list\_node\_base \* **\_M\_erase\_after** (\_Fwd\_list\_node\_base \* \_\_pos)
- \_Fwd\_list\_node\_base \* **\_M\_erase\_after** (\_Fwd\_list\_node\_base \* \_\_pos, \_Fwd\_list\_node\_base \* \_\_last)
- \_Node \* **\_M\_get\_node** ()
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept
- template<typename... \_Args>  
\_Fwd\_list\_node\_base \* **\_M\_insert\_after** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- void **\_M\_put\_node** (\_Node \* \_\_p)

#### Private Attributes

- \_Fwd\_list\_impl **\_M\_impl**

#### 4.707.1 Detailed Description

template<typename \_Tp, typename \_Alloc = allocator<\_Tp>>class std::forward\_list< \_Tp, \_Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

#### Template Parameters

|               |                                             |
|---------------|---------------------------------------------|
| <b>_Tp</b>    | Type of element.                            |
| <b>_Alloc</b> | Allocator type, defaults to allocator<_Tp>. |

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

Definition at line 408 of file `forward_list.h`.

## 4.707.2 Constructor &amp; Destructor Documentation

4.707.2.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( const _Alloc & __al = _Alloc() ) [inline], [explicit]`

Creates a forward\_list with no elements.

## Parameters

|                   |                      |
|-------------------|----------------------|
| <code>__al</code> | An allocator object. |
|-------------------|----------------------|

Definition at line 440 of file forward\_list.h.

4.707.2.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( const forward_list< _Tp, _Alloc > & __list, const _Alloc & __al ) [inline]`

Copy constructor with allocator argument.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__list</code> | Input list to copy.  |
| <code>__al</code>   | An allocator object. |

Definition at line 449 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::begin(), and std::forward\_list< \_Tp, \_Alloc >::end().

4.707.2.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( forward_list< _Tp, _Alloc > && __list, const _Alloc & __al ) [inline], [noexcept]`

Move constructor with allocator argument.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__list</code> | Input list to move.  |
| <code>__al</code>   | An allocator object. |

Definition at line 458 of file forward\_list.h.

4.707.2.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n, const _Alloc & __al = _Alloc() ) [inline], [explicit]`

Creates a forward\_list with default constructed elements.

## Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
|------------------|---------------------------------------------|

This constructor creates the forward\_list with `__n` default constructed elements.

Definition at line 471 of file forward\_list.h.

4.707.2.5 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n, const _Tp & __value, const _Alloc & __al = _Alloc() ) [inline]`

Creates a forward\_list with copies of an exemplar element.

## Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__al</code>    | An allocator object.                        |

This constructor fills the forward\_list with `__n` copies of `__value`.

Definition at line 484 of file forward\_list.h.

```
4.707.2.6 template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename =
std::RequireInputIter<_InputIterator>> std::forward_list< _Tp, _Alloc >::forward_list (_InputIterator __first,
_InputIterator __last, const _Alloc & __al = _Alloc()) [inline]
```

Builds a forward\_list from a range.

## Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__al</code>    | An allocator object. |

Create a forward\_list consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is distance(`__first`,`__last`)).

Definition at line 501 of file forward\_list.h.

```
4.707.2.7 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list (
const forward_list< _Tp, _Alloc > & __list) [inline]
```

The forward\_list copy constructor.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

Definition at line 511 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::begin(), and std::forward\_list< \_Tp, \_Alloc >::end().

```
4.707.2.8 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list (
forward_list< _Tp, _Alloc > && __list) [inline], [noexcept]
```

The forward\_list move constructor.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The newly-created forward\_list contains the exact contents of `__list`. The contents of `__list` are a valid, but unspecified forward\_list.

Definition at line 525 of file forward\_list.h.

```
4.707.2.9 template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::forward_list (
std::initializer_list< _Tp > __il, const _Alloc & __al = _Alloc()) [inline]
```

Builds a forward\_list from an initializer\_list.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
| <code>__al</code> | An allocator object.               |

Create a forward\_list consisting of copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

Definition at line 536 of file forward\_list.h.

**4.707.2.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list< _Tp, _Alloc >::~forward_list ( ) [inline],[noexcept]`

The forward\_list dtor.

Definition at line 544 of file forward\_list.h.

**4.707.3 Member Function Documentation**

**4.707.3.1** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator , typename = std::RequireInputIter<_InputIterator>> void std::forward_list< _Tp, _Alloc >::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Assigns a range to a forward\_list.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a forward\_list with copies of the elements in the range `[ __first, __last )`.

Note that the assignment completely changes the forward\_list and that the number of elements of the resulting forward\_list is the same as the number of elements assigned. Old data is lost.

Definition at line 609 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::operator=()`.

**4.707.3.2** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::assign ( size_type __n, const _Tp & __val ) [inline]`

Assigns a given value to a forward\_list.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a forward\_list with `__n` copies of the given value. Note that the assignment completely changes the forward\_list, and that the resulting forward\_list has `__n` elements. Old data is lost.

Definition at line 626 of file forward\_list.h.

**4.707.3.3** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::assign ( std::initializer_list<_Tp> __il ) [inline]`

Assigns an initializer\_list to a forward\_list.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
|-------------------|------------------------------------|

Replace the contents of the forward\_list with copies of the elements in the initializer\_list `__il`. This is linear in `il.size()`.

Definition at line 638 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::assign()`.

Referenced by `std::forward_list< _Tp, _Alloc >::assign()`.

**4.707.3.4** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 653 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

**4.707.3.5** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 662 of file `forward_list.h`.

**4.707.3.6** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 670 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`.

**4.707.3.7** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 679 of file `forward_list.h`.

**4.707.3.8** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 715 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::emplace_front()`, and `std::forward_list< _Tp, _Alloc >::push_front()`.

**4.707.3.9** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 706 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==( ).

**4.707.3.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 724 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==( ).

**4.707.3.11** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::clear ( ) [inline], [noexcept]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1027 of file forward\_list.h.

**4.707.3.12** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> iterator std::forward_list< _Tp, _Alloc >::emplace_after ( const_iterator __pos, _Args &&... __args ) [inline]`

Constructs object in forward\_list after the specified iterator.

#### Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos</code>  | A const_iterator into the forward_list. |
| <code>__args</code> | Arguments.                              |

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 837 of file forward\_list.h.

**4.707.3.13** `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> void std::forward_list< _Tp, _Alloc >::emplace_front ( _Args &&... __args ) [inline]`

Constructs object in forward\_list at the front of the list.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__args</code> | Arguments. |
|---------------------|------------|

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args)...)`  at the front of the list. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 781 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`.

**4.707.3.14** `template<typename _Tp, typename _Alloc = allocator<_Tp>> bool std::forward_list< _Tp, _Alloc >::empty ( )`  
`const [inline], [noexcept]`

Returns true if the `forward_list` is empty. (Thus `begin()` would equal `end()`.)

Definition at line 732 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

**4.707.3.15** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::end ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 688 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`, and `std::forward_list< _Tp, _Alloc >::insert_after()`.

**4.707.3.16** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::end ( ) const` `[inline], [noexcept]`

Returns a read-only iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 697 of file `forward_list.h`.

**4.707.3.17** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::erase_after ( const_iterator __pos ) [inline]`

Removes the element pointed to by the iterator following `pos`.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Iterator pointing before element to be erased. |
|--------------------|------------------------------------------------|

#### Returns

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 940 of file `forward_list.h`.

**4.707.3.18** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::erase_after ( const_iterator __pos, const_iterator __last ) [inline]`

Remove a range of elements.

## Parameters

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator pointing before the first element to be erased.     |
| <code>__last</code> | Iterator pointing to one past the last element to be erased. |

## Returns

@ `__last`.

This function will erase the elements in the range (`__pos,__last`) and shorten the `forward_list` accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 963 of file `forward_list.h`.

**4.707.3.19** `template<typename _Tp, typename _Alloc = allocator<_Tp>> reference std::forward_list< _Tp, _Alloc >::front ( )`  
`[inline]`

Returns a read/write reference to the data at the first element of the `forward_list`.

Definition at line 749 of file `forward_list.h`.

**4.707.3.20** `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_reference std::forward_list< _Tp, _Alloc >::front ( ) const` `[inline]`

Returns a read-only (constant) reference to the data at the first element of the `forward_list`.

Definition at line 760 of file `forward_list.h`.

**4.707.3.21** `template<typename _Tp, typename _Alloc = allocator<_Tp>> allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ( ) const` `[inline], [noexcept]`

Get a copy of the memory allocation object.

Definition at line 643 of file `forward_list.h`.

**4.707.3.22** `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::insert_after ( const_iterator __pos, const _Tp & __val )` `[inline]`

Inserts given value into `forward_list` after specified iterator.

## Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> . |
| <code>__val</code> | Data to be inserted.                             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 854 of file forward\_list.h.

**4.707.3.23** `template<typename _Tp, typename _Alloc > forward_list< _Tp, _Alloc >::iterator forward_list::insert_after ( const_iterator __pos, size_type __n, const _Tp & __val )`

Inserts a number of copies of given data into the forward\_list.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__pos</code> | An iterator into the forward_list. |
| <code>__n</code>   | Number of elements to be inserted. |
| <code>__val</code> | Data to be inserted.               |

**Returns**

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 272 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::before\_begin(), and std::forward\_list< \_Tp, \_Alloc >::end().

**4.707.3.24** `template<typename _Tp, typename _Alloc > template<typename _InputIterator, typename > forward_list< _Tp, _Alloc >::iterator forward_list::insert_after ( const_iterator __pos, _InputIterator __first, _InputIterator __last )`

Inserts a range into the forward\_list.

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>__pos</code>   | An iterator into the forward_list. |
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |

**Returns**

An iterator pointing to the last inserted element or *\_\_pos* if *\_\_first* == *\_\_last*.

This function will insert copies of the data in the range [*\_\_first*, *\_\_last*) into the forward\_list after the location specified by *\_\_pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 287 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::before\_begin(), std::forward\_list< \_Tp, \_Alloc >::empty(), and std::forward\_list< \_Tp, \_Alloc >::end().

4.707.3.25 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::insert_after ( const_iterator __pos, std::initializer_list< _Tp > __il ) [inline]`

Inserts the contents of an initializer\_list into forward\_list after the specified iterator.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__pos</code> | An iterator into the forward_list. |
| <code>__il</code>  | An initializer_list of value_type. |

#### Returns

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the initializer\_list `__il` into the forward\_list before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 919 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::insert_after()`.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

4.707.3.26 `template<typename _Tp, typename _Alloc = allocator<_Tp>> size_type std::forward_list< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]`

Returns the largest possible number of elements of forward\_list.

Definition at line 739 of file forward\_list.h.

References `std::allocator_traits< _Alloc >::max_size()`.

4.707.3.27 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::merge ( forward_list< _Tp, _Alloc > && __list ) [inline]`

Merge sorted lists.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__list</code> | Sorted list to merge. |
|---------------------|-----------------------|

Assumes that both `list` and this list are sorted according to operator<(). Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Definition at line 1165 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::merge()`.

Referenced by `std::forward_list< _Tp, _Alloc >::merge()`.

4.707.3.28 `template<typename _Tp, typename _Alloc > template<typename _Comp > void forward_list::merge ( forward_list< _Tp, _Alloc > && __list, _Comp __comp )`

Merge sorted lists according to comparison function.

#### Parameters

|                     |                                          |
|---------------------|------------------------------------------|
| <code>__list</code> | Sorted list to merge.                    |
| <code>__comp</code> | Comparison function defining sort order. |

Assumes that both `__list` and this list are sorted according to `comp`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to `comp()`.

Definition at line 365 of file `forward_list.tcc`.

**4.707.3.29** `template<typename _Tp, typename _Alloc > forward_list< _Tp, _Alloc > & forward_list::operator= ( const forward_list< _Tp, _Alloc > & __list )`

The `forward_list` assignment operator.

#### Parameters

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__list</code> | A <code>forward_list</code> of identical element and allocator types. |
|---------------------|-----------------------------------------------------------------------|

All the elements of `__list` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 156 of file `forward_list.tcc`.

References `std::forward_list< _Tp, _Alloc >::cbegin()`, and `std::forward_list< _Tp, _Alloc >::cend()`.

**4.707.3.30** `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list< _Tp, _Alloc >::operator= ( forward_list< _Tp, _Alloc > && __list ) [inline], [noexcept]`

The `forward_list` move assignment operator.

#### Parameters

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__list</code> | A <code>forward_list</code> of identical element and allocator types. |
|---------------------|-----------------------------------------------------------------------|

The contents of `__list` are moved into this `forward_list` (without copying, if the allocators permit it). `__list` is a valid, but unspecified `forward_list`

Definition at line 568 of file `forward_list.h`.

**4.707.3.31** `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list< _Tp, _Alloc >::operator= ( std::initializer_list< _Tp > __il ) [inline]`

The `forward_list` initializer list assignment operator.

#### Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__il</code> | An <code>initializer_list</code> of <code>value_type</code> . |
|-------------------|---------------------------------------------------------------|

Replace the contents of the `forward_list` with copies of the elements in the `initializer_list` `__il`. This is linear in `__il.size()`.

Definition at line 588 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::assign()`.

**4.707.3.32** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the `forward_list` by one. Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 819 of file `forward_list.h`.

4.707.333 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::push_front ( const _Tp & __val ) [inline]`

Add data to the front of the forward\_list.

#### Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__val</code> | Data to be added. |
|--------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the forward\_list and assigns the given data to it. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 796 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`.

4.707.334 `template<typename _Tp, typename _Alloc > void forward_list::remove ( const _Tp & __val )`

Remove all elements equal to value.

#### Parameters

|                    |                      |
|--------------------|----------------------|
| <code>__val</code> | The value to remove. |
|--------------------|----------------------|

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 300 of file forward\_list.tcc.

References `std::__addressof()`.

4.707.335 `template<typename _Tp, typename _Alloc > template<typename _Pred > void forward_list::remove_if ( _Pred __pred )`

Remove all elements satisfying a predicate.

#### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__pred</code> | Unary predicate function or object. |
|---------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 328 of file forward\_list.tcc.

4.707.336 `template<typename _Tp, typename _Alloc > void forward_list::resize ( size_type __sz )`

Resizes the forward\_list to the specified number of elements.

#### Parameters

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__sz</code> | Number of elements the forward_list should contain. |
|-------------------|-----------------------------------------------------|

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current number of elements the forward\_list is truncated, otherwise the forward\_list is extended and the new elements are default constructed.

Definition at line 198 of file forward\_list.tcc.

References std::end().

**4.707.337** template<typename \_Tp, typename \_Alloc > void forward\_list::resize ( size\_type \_\_sz, const value\_type & \_\_val )

Resizes the forward\_list to the specified number of elements.

#### Parameters

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__sz</code>  | Number of elements the forward_list should contain. |
| <code>__val</code> | Data with which new elements should be populated.   |

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current number of elements the forward\_list is truncated, otherwise the forward\_list is extended and new elements are populated with given data.

Definition at line 217 of file forward\_list.tcc.

References std::end().

**4.707.338** template<typename \_Tp, typename \_Alloc = allocator<\_Tp>> void std::forward\_list< \_Tp, \_Alloc >::reverse ( )  
[inline], [noexcept]

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1218 of file forward\_list.h.

**4.707.339** template<typename \_Tp, typename \_Alloc = allocator<\_Tp>> void std::forward\_list< \_Tp, \_Alloc >::sort ( )  
[inline]

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1199 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::sort().

Referenced by std::forward\_list< \_Tp, \_Alloc >::sort().

**4.707.340** template<typename \_Tp, class \_Alloc > template<typename \_Comp > void forward\_list::sort ( \_Comp \_\_comp )

Sort the forward\_list using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 411 of file forward\_list.tcc.

**4.707.341** template<typename \_Tp, typename \_Alloc = allocator<\_Tp>> void std::forward\_list< \_Tp, \_Alloc >::splice\_after ( const\_iterator \_\_pos, forward\_list< \_Tp, \_Alloc > && \_\_list ) [inline]

Insert contents of another forward\_list.

#### Parameters

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after. |
| <code>__list</code> | Source list.                                      |

The elements of *list* are inserted in constant time after the element referenced by *pos*. *list* becomes an empty list.

Requires this != x.

Definition at line 1044 of file forward\_list.h.

**4.707.3.42** `template<typename _Tp, typename _Alloc > void forward_list::splice_after ( const_iterator __pos, forward_list< _Tp, _Alloc > && __list, const_iterator __i )`

Insert element from another forward\_list.

#### Parameters

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after.            |
| <code>__list</code> | Source list.                                                 |
| <code>__i</code>    | Iterator referencing the element before the element to move. |

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 255 of file forward\_list.tcc.

**4.707.3.43** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after ( const_iterator __pos, forward_list< _Tp, _Alloc > && , const_iterator __before, const_iterator __last ) [inline]`

Insert range from another forward\_list.

#### Parameters

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.       |
| <code>__list</code>   | Source list.                                            |
| <code>__before</code> | Iterator referencing before the start of range in list. |
| <code>__last</code>   | Iterator referencing the end of range in list.          |

Removes elements in the range (`__before`,`__last`) and inserts them after `__pos` in constant time.

Undefined if `__pos` is in (`__before`,`__last`).

Definition at line 1087 of file forward\_list.h.

**4.707.3.44** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::swap ( forward_list< _Tp, _Alloc > & __list ) [inline], [noexcept]`

Swaps data with another forward\_list.

#### Parameters

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__list</code> | A forward_list of the same element and allocator types. |
|---------------------|---------------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 980 of file forward\_list.h.

Referenced by `std::swap()`.

**4.707.3.45** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::unique ( ) [inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1136 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::unique()`.

Referenced by `std::forward_list<_Tp, _Alloc>::unique()`.

**4.707.3.46** `template<typename _Tp, typename _Alloc> template<typename _BinPred> void forward_list::unique ( _BinPred __binary_pred )`

Remove consecutive elements satisfying a predicate.

#### Parameters

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>__binary_pred</code> | Binary predicate function or object. |
|----------------------------|--------------------------------------|

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 344 of file `forward_list.tcc`.

References `std::begin()`, and `std::end()`.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 4.708 `std::fpos<_StateT>` Class Template Reference

### Public Member Functions

- [fpos \(streamoff \\_\\_off\)](#)
- [operator streamoff \(\) const](#)
- [fpos operator+ \(streamoff \\_\\_off\) const](#)
- [fpos & operator+= \(streamoff \\_\\_off\)](#)
- [fpos operator- \(streamoff \\_\\_off\) const](#)
- [streamoff operator- \(const fpos & \\_\\_other\) const](#)
- [fpos & operator-= \(streamoff \\_\\_off\)](#)
- [void state \(\\_StateT \\_\\_st\)](#)
- [\\_StateT state \(\) const](#)

### 4.708.1 Detailed Description

`template<typename _StateT> class std::fpos<_StateT>`

Class representing stream positions.

The standard places no requirements upon the template parameter `StateT`. In this implementation `StateT` must be `DefaultConstructible`, `CopyConstructible` and `Assignable`. The standard only requires that `fpos` should contain a member of type `StateT`. In this implementation it also contains an offset stored as a signed integer.

## Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>StateT</i> | Type passed to and returned from state(). |
|---------------|-------------------------------------------|

Definition at line 112 of file postypes.h.

## 4.708.2 Constructor &amp; Destructor Documentation

## 4.708.2.1 template&lt;typename \_StateT&gt; std::fpos&lt;\_StateT&gt;::fpos ( streamoff \_\_off ) [inline]

Construct position from offset.

Definition at line 133 of file postypes.h.

## 4.708.3 Member Function Documentation

## 4.708.3.1 template&lt;typename \_StateT&gt; std::fpos&lt;\_StateT&gt;::operator streamoff ( ) const [inline]

Convert to streamoff.

Definition at line 137 of file postypes.h.

## 4.708.3.2 template&lt;typename \_StateT&gt; fpos std::fpos&lt;\_StateT&gt;::operator+ ( streamoff \_\_off ) const [inline]

Add position and offset.

Definition at line 178 of file postypes.h.

## 4.708.3.3 template&lt;typename \_StateT&gt; fpos&amp; std::fpos&lt;\_StateT&gt;::operator+= ( streamoff \_\_off ) [inline]

Add offset to this position.

Definition at line 154 of file postypes.h.

## 4.708.3.4 template&lt;typename \_StateT&gt; fpos std::fpos&lt;\_StateT&gt;::operator- ( streamoff \_\_off ) const [inline]

Subtract offset from position.

Definition at line 192 of file postypes.h.

## 4.708.3.5 template&lt;typename \_StateT&gt; streamoff std::fpos&lt;\_StateT&gt;::operator- ( const fpos&lt;\_StateT&gt; &amp; \_\_other ) const [inline]

Subtract position to return offset.

Definition at line 205 of file postypes.h.

## 4.708.3.6 template&lt;typename \_StateT&gt; fpos&amp; std::fpos&lt;\_StateT&gt;::operator-= ( streamoff \_\_off ) [inline]

Subtract offset from this position.

Definition at line 165 of file postypes.h.

## 4.708.3.7 template&lt;typename \_StateT&gt; void std::fpos&lt;\_StateT&gt;::state ( \_StateT \_\_st ) [inline]

Remember the value of *st*.

Definition at line 141 of file postypes.h.

4.708.3.8 `template<typename _StateT> _StateT std::fpos<_StateT>::state ( ) const [inline]`

Return the last set value of *st*.

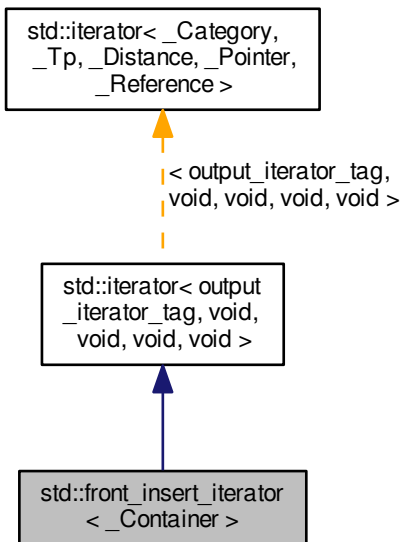
Definition at line 146 of file `postypes.h`.

The documentation for this class was generated from the following file:

- [postypes.h](#)

## 4.709 std::front\_insert\_iterator<\_Container> Class Template Reference

Inheritance diagram for `std::front_insert_iterator<_Container>`:



### Public Types

- `typedef _Container container_type`
- `typedef void difference_type`
- `typedef output_iterator_tag iterator_category`
- `typedef void pointer`
- `typedef void reference`
- `typedef void value_type`

### Public Member Functions

- `front_insert_iterator (_Container &__x)`
- `front_insert_iterator & operator* ()`

- [front\\_insert\\_iterator](#) & [operator++](#) ()
- [front\\_insert\\_iterator](#) [operator++](#) (int)
- [front\\_insert\\_iterator](#) & [operator=](#) (const typename \_Container::value\_type &\_\_value)
- [front\\_insert\\_iterator](#) & [operator=](#) (typename \_Container::value\_type &&\_\_value)

#### Protected Attributes

- \_Container \* **container**

#### 4.709.1 Detailed Description

```
template<typename _Container>class std::front_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 493 of file `stl_iterator.h`.

#### 4.709.2 Member Typedef Documentation

4.709.2.1 `template<typename _Container> typedef _Container std::front_insert_iterator< _Container >::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 501 of file `stl_iterator.h`.

4.709.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.709.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.709.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` `[inherited]`

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.709.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference` `[inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.709.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type` `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

#### 4.709.3 Constructor & Destructor Documentation

4.709.3.1 `template<typename _Container> std::front_insert_iterator<_Container>::front_insert_iterator ( _Container & __x ) [inline], [explicit]`

The only way to create this iterator is with a container.

Definition at line 504 of file stl\_iterator.h.

#### 4.709.4 Member Function Documentation

4.709.4.1 `template<typename _Container> front_insert_iterator& std::front_insert_iterator<_Container>::operator* ( ) [inline]`

Simply returns `*this`.

Definition at line 542 of file stl\_iterator.h.

4.709.4.2 `template<typename _Container> front_insert_iterator& std::front_insert_iterator<_Container>::operator++ ( ) [inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 547 of file stl\_iterator.h.

4.709.4.3 `template<typename _Container> front_insert_iterator std::front_insert_iterator<_Container>::operator++ ( int ) [inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 552 of file stl\_iterator.h.

4.709.4.4 `template<typename _Container> front_insert_iterator& std::front_insert_iterator<_Container>::operator= ( const typename _Container::value_type & __value ) [inline]`

##### Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

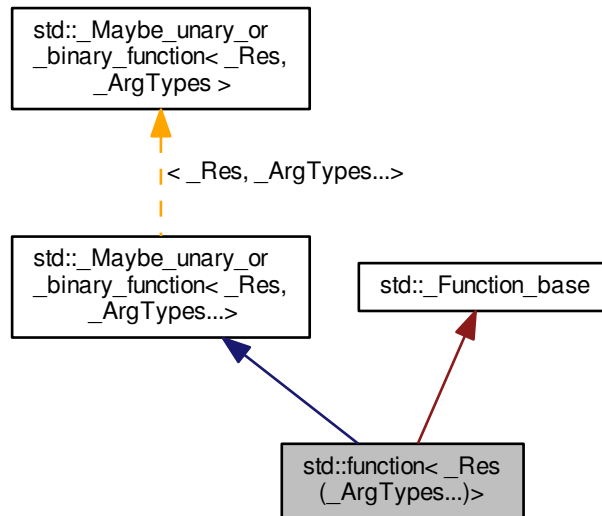
Definition at line 526 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.710 std::function&lt; \_Res(\_ArgTypes...)&gt; Class Template Reference

Inheritance diagram for std::function< \_Res(\_ArgTypes...)>:



## Public Types

- typedef `_Res` **result\_type**

## Public Member Functions

- `function` () noexcept
- `function` (nullptr\_t) noexcept
- `function` (const function &\_\_x)
- `function` (function &&\_\_x)
- template<typename \_Functor, typename = \_Requires<\_Callable<\_Functor>, void>>  
  `function` (\_Functor)
- `operator bool` () const noexcept
- `_Res operator()` (\_ArgTypes... \_\_args) const
- `function` & `operator=` (const function &\_\_x)
- `function` & `operator=` (function &&\_\_x)
- `function` & `operator=` (nullptr\_t)
- template<typename \_Functor >  
  \_requires<\_Callable<\_Functor>  
  , function & > `operator=` (\_Functor &&\_\_f)
- template<typename \_Functor >  
  `function` & `operator=` (reference\_wrapper<\_Functor> \_\_f) noexcept
- void `swap` (function &\_\_x)

- template<typename \_Functor >  
\_Functor \* [target](#) () noexcept
- template<typename \_Functor >  
const \_Functor \* [target](#) () const noexcept
- const [type\\_info](#) & [target\\_type](#) () const noexcept

#### Private Types

- typedef bool(\* **\_Manager\_type**)(\_Any\_data &, const \_Any\_data &, \_Manager\_operation)

#### Private Member Functions

- bool **\_M\_empty** () const

#### Private Attributes

- \_Any\_data **\_M\_functor**
- \_Manager\_type **\_M\_manager**

#### Static Private Attributes

- static const std::size\_t **\_M\_max\_align**
- static const std::size\_t **\_M\_max\_size**

#### 4.710.1 Detailed Description

template<typename \_Res, typename... \_ArgTypes> class std::function< \_Res(\_ArgTypes...)>

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 2170 of file functional.

#### 4.710.2 Constructor & Destructor Documentation

4.710.2.1 template<typename \_Res, typename... \_ArgTypes> std::function< \_Res(\_ArgTypes...)>::function ( ) [inline],  
[noexcept]

Default construct creates an empty function call wrapper.

#### Postcondition

!(bool)\*this

Definition at line 2203 of file functional.

4.710.2.2 `template<typename _Res, typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( nullptr_t )`  
`[inline], [noexcept]`

Creates an empty function call wrapper.

#### Postcondition

`!(bool)*this`

Definition at line 2210 of file functional.

4.710.2.3 `template<typename _Res, typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( const function< _Res(_ArgTypes...)> & __x )`

Function copy constructor.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A function object with identical call signature. |
|------------------|--------------------------------------------------|

#### Postcondition

`bool(*this) == bool(__x)`

The newly-created function contains a copy of the target of `__x` (if it has one).

Definition at line 2434 of file functional.

4.710.2.4 `template<typename _Res, typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( function< _Res(_ArgTypes...)> && __x )` `[inline]`

Function move constructor.

#### Parameters

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>__x</code> | A function object rvalue with identical call signature. |
|------------------|---------------------------------------------------------|

The newly-created function contains the target of `__x` (if it has one).

Definition at line 2230 of file functional.

4.710.2.5 `template<typename _Res, typename... _ArgTypes> template<typename _Functor, typename > std::function< _Res(_ArgTypes...)>::function ( _Functor __f )`

Builds a function that targets a copy of the incoming function object.

#### Parameters

|                  |                                                                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__f</code> | A function object that is callable with parameters of type <code>T1, T2, ..., TN</code> and returns a value convertible to <code>Res</code> . |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

The newly-created function object will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If `__f` is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2448 of file functional.

## 4.710.3 Member Function Documentation

4.710.3.1 `template<typename _Res, typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::operator bool ( ) const`  
`[inline], [explicit], [noexcept]`

Determine if the function wrapper has a target.

## Returns

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 2383 of file functional.

4.710.3.2 `template<typename _Res, typename... _ArgTypes> _Res std::function< _Res(_ArgTypes...)>::operator() ( _ArgTypes...  
 __args ) const`

Invokes the function targeted by `*this`.

## Returns

the result of the target.

## Exceptions

|                                |                                |
|--------------------------------|--------------------------------|
| <code>bad_function_call</code> | when <code>!(bool)*this</code> |
|--------------------------------|--------------------------------|

The function call operator invokes the target function object stored by `this`.

Definition at line 2464 of file functional.

4.710.3.3 `template<typename _Res, typename... _ArgTypes> function& std::function< _Res(_ArgTypes...)>::operator= ( const  
 function< _Res(_ArgTypes...)> & __x ) [inline]`

Function assignment operator.

## Parameters

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__x</code> | A function with identical call signature. |
|------------------|-------------------------------------------|

## Postcondition

`(bool)*this == (bool)x`

## Returns

`*this`

The target of `__x` is copied to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 2270 of file functional.

4.710.3.4 `template<typename _Res, typename... _ArgTypes> function& std::function<_Res(_ArgTypes...)>::operator= (function<_Res(_ArgTypes...)> && __x ) [inline]`

Function move-assignment operator.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A function rvalue with identical call signature. |
|------------------|--------------------------------------------------|

#### Returns

`*this`

The target of `__x` is moved to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 2288 of file functional.

4.710.3.5 `template<typename _Res, typename... _ArgTypes> function& std::function<_Res(_ArgTypes...)>::operator= (nullptr ) [inline]`

Function assignment to zero.

#### Postcondition

`!(bool)*this`

#### Returns

`*this`

The target of `*this` is deallocated, leaving it empty.

Definition at line 2302 of file functional.

4.710.3.6 `template<typename _Res, typename... _ArgTypes> template<typename _Functor> _Requires<Callable<_Functor>, function&> std::function<_Res(_ArgTypes...)>::operator= ( _Functor && __f ) [inline]`

Function assignment to a new target.

#### Parameters

|                  |                                                                                                                                  |
|------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <code>__f</code> | A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to <code>Res</code> . |
|------------------|----------------------------------------------------------------------------------------------------------------------------------|

#### Returns

`*this`

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If `__f` is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2331 of file functional.

4.710.3.7 `template<typename _Res, typename... _ArgTypes> template<typename _Functor > function& std::function< _Res(_ArgTypes...)>::operator= ( reference_wrapper< _Functor > __f ) [inline], [noexcept]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2340 of file functional.

4.710.3.8 `template<typename _Res, typename... _ArgTypes> void std::function< _Res(_ArgTypes...)>::swap ( function< _Res(_ArgTypes...)> & __x ) [inline]`

Swap the targets of two function objects.

#### Parameters

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__x</code> | A function with identical call signature. |
|------------------|-------------------------------------------|

Swap the targets of `this` function object and `__f`. This function will not throw an exception.

Definition at line 2355 of file functional.

4.710.3.9 `template<typename _Res, typename... _ArgTypes> template<typename _Functor > _Functor * std::function< _Res(_ArgTypes...)>::target ( ) [noexcept]`

Access the stored target function object.

#### Returns

Returns a pointer to the stored target function object, if `typeid(Functor).equals(target_type())`; otherwise, a NULL pointer.

This function will not throw an exception.

Definition at line 2491 of file functional.

4.710.3.10 `template<typename _Res, typename... _ArgTypes> template<typename _Functor > const _Functor * std::function< _Res(_ArgTypes...)>::target ( ) const [noexcept]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2510 of file functional.

4.710.3.11 `template<typename _Res, typename... _ArgTypes> const type_info & std::function< _Res(_ArgTypes...)>::target_type ( ) const [noexcept]`

Determine the type of the target of this function object wrapper.

#### Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

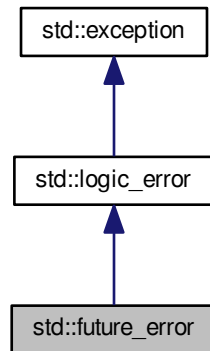
Definition at line 2475 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 4.711 `std::future_error` Class Reference

Inheritance diagram for `std::future_error`:



### Public Member Functions

- **future\_error** ([error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const noexcept
- virtual const char \* **what** () const noexcept

#### 4.711.1 Detailed Description

Exception type thrown by futures.

Definition at line 93 of file future.

#### 4.711.2 Member Function Documentation

##### 4.711.2.1 virtual const char\* `std::future_error::what ( ) const` [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

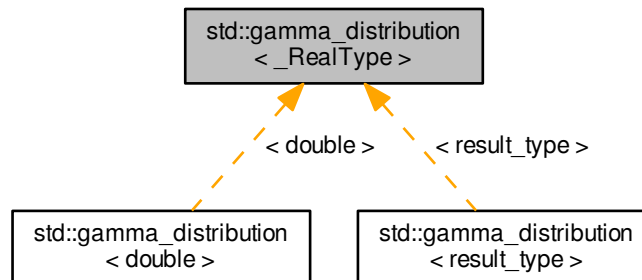
Reimplemented from [std::logic\\_error](#).

The documentation for this class was generated from the following file:

- [future](#)

## 4.712 std::gamma\_distribution&lt;\_RealType&gt; Class Template Reference

Inheritance diagram for std::gamma\_distribution<\_RealType>:



## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [gamma\\_distribution](#) (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- **gamma\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_RealType alpha` () const
- `_RealType beta` () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
std::basic\_ostream<\_CharT,  
\_Traits> & operator<< (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const std::gamma\_distribution<\_RealType1> &\_\_x)
- bool operator== (const gamma\_distribution &\_\_d1, const gamma\_distribution &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
std::basic\_istream<\_CharT,  
\_Traits> & operator>> (std::basic\_istream<\_CharT, \_Traits> &\_\_is, std::gamma\_distribution<\_RealType1> &\_\_x)

## 4.712.1 Detailed Description

template<typename \_RealType = double> class std::gamma\_distribution<\_RealType>

A gamma continuous distribution for random numbers.

The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta \Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2502 of file random.h.

## 4.712.2 Member Typedef Documentation

4.712.2.1 template<typename \_RealType = double> typedef \_RealType std::gamma\_distribution<\_RealType>::result\_type

The type of the range of the distribution.

Definition at line 2505 of file random.h.

## 4.712.3 Constructor &amp; Destructor Documentation

4.712.3.1 template<typename \_RealType = double> std::gamma\_distribution<\_RealType>::gamma\_distribution  
( \_RealType \_\_alpha\_val = \_RealType(1), \_RealType \_\_beta\_val = \_RealType(1) ) [inline],  
[explicit]

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

Definition at line 2554 of file random.h.

## 4.712.4 Member Function Documentation

4.712.4.1 template<typename \_RealType = double> \_RealType std::gamma\_distribution<\_RealType>::alpha( ) const  
[inline]

Returns the  $\alpha$  of the distribution.

Definition at line 2575 of file random.h.

4.712.4.2 `template<typename _RealType = double> _RealType std::gamma_distribution<_RealType>::beta ( ) const`  
`[inline]`

Returns the  $\beta$  of the distribution.

Definition at line 2582 of file random.h.

4.712.4.3 `template<typename _RealType = double> result_type std::gamma_distribution<_RealType>::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2611 of file random.h.

4.712.4.4 `template<typename _RealType = double> result_type std::gamma_distribution<_RealType>::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2604 of file random.h.

4.712.4.5 `template<typename _RealType = double> template<typename UniformRandomNumberGenerator> result_type`  
`std::gamma_distribution<_RealType>::operator() ( UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2619 of file random.h.

Referenced by std::gamma\_distribution<result\_type>::operator()().

4.712.4.6 `template<typename _RealType> template<typename UniformRandomNumberGenerator> gamma_distribution<`  
`_RealType>::result_type std::gamma_distribution<_RealType>::operator() ( UniformRandomNumberGenerator`  
`& __urng, const param_type & __param )`

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2487 of file bits/random.tcc.

References std::log(), and std::pow().

4.712.4.7 `template<typename _RealType = double> param_type std::gamma_distribution<_RealType>::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 2589 of file random.h.

4.712.4.8 `template<typename _RealType = double> void std::gamma_distribution<_RealType>::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2597 of file random.h.

4.712.4.9 `template<typename _RealType = double> void std::gamma_distribution<_RealType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 2568 of file random.h.

Referenced by `std::chi_squared_distribution<_RealType>::reset()`, `std::fisher_f_distribution<_RealType>::reset()`, `std::student_t_distribution<_RealType>::reset()`, and `std::negative_binomial_distribution<_IntType>::reset()`.

#### 4.712.5 Friends And Related Function Documentation

4.712.5.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<<( const std::basic_ostream<_CharT, _Traits> &__os, const std::gamma_distribution<_RealType1> &__x ) [friend]`

Inserts a gamma\_distribution random number distribution \_\_x into the output stream \_\_os.

##### Parameters

|                   |                                                  |
|-------------------|--------------------------------------------------|
| <code>__os</code> | An output stream.                                |
| <code>__x</code>  | A gamma_distribution random number distribution. |

##### Returns

The output stream with the state of \_\_x inserted or in an error state.

4.712.5.2 `template<typename _RealType = double> bool operator==( const gamma_distribution<_RealType> &__d1, const gamma_distribution<_RealType> &__d2 ) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2655 of file random.h.

4.712.5.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> > std::basic_istream<_CharT, _Traits>& operator>>( std::basic_istream<_CharT, _Traits> &__is, std::gamma_distribution<_RealType1> &__x ) [friend]`

Extracts a gamma\_distribution random number distribution \_\_x from the input stream \_\_is.

##### Parameters

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <code>__is</code> | An input stream.                                     |
| <code>__x</code>  | A gamma_distribution random number generator engine. |

##### Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.713 `std::gamma_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef [gamma\\_distribution](#)  
`<_RealType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- `_RealType` **alpha** () const
- `_RealType` **beta** () const

## Friends

- class **gamma\_distribution**`<_RealType>`
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.713.1 Detailed Description

`template<typename _RealType = double>struct std::gamma_distribution<_RealType>::param_type`

Parameter type.

Definition at line 2511 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.714 `std::geometric_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- **geometric\_distribution** (`double __p=0.5`)
- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
`void` **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
`void` **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const`  
[param\\_type](#) &\_\_p)

- `template<typename _UniformRandomNumberGenerator >`  
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `double p () const`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `bool operator== (const geometric_distribution &__d1, const geometric_distribution &__d2)`

#### 4.714.1 Detailed Description

`template<typename _IntType = int> class std::geometric_distribution<_IntType>`

A discrete geometric random number distribution.

The formula for the geometric probability density function is  $p(i|p) = p(1-p)^i$  where  $p$  is the parameter of the distribution.

Definition at line 4007 of file random.h.

#### 4.714.2 Member Typedef Documentation

4.714.2.1 `template<typename _IntType = int> typedef _IntType std::geometric_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 4010 of file random.h.

#### 4.714.3 Member Function Documentation

4.714.3.1 `template<typename _IntType = int> result_type std::geometric_distribution<_IntType>::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4099 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

4.714.3.2 `template<typename _IntType = int> result_type std::geometric_distribution<_IntType>::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4092 of file random.h.

4.714.3.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type  
std::geometric_distribution< _IntType >::operator()( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4107 of file random.h.

References std::geometric\_distribution< \_IntType >::operator()().

Referenced by std::geometric\_distribution< \_IntType >::operator()().

4.714.3.4 `template<typename _IntType = int> double std::geometric_distribution< _IntType >::p ( ) const [inline]`

Returns the distribution parameter p.

Definition at line 4070 of file random.h.

4.714.3.5 `template<typename _IntType = int> param_type std::geometric_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4077 of file random.h.

Referenced by std::operator>>().

4.714.3.6 `template<typename _IntType = int> void std::geometric_distribution< _IntType >::param ( const param_type &  
__param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4085 of file random.h.

4.714.3.7 `template<typename _IntType = int> void std::geometric_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 4064 of file random.h.

#### 4.714.4 Friends And Related Function Documentation

4.714.4.1 `template<typename _IntType = int> bool operator== ( const geometric_distribution< _IntType > & __d1, const  
geometric_distribution< _IntType > & __d2 ) [friend]`

Return true if two geometric distributions have the same parameters.

Definition at line 4142 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.715 `std::geometric_distribution<_IntType>::param_type` Struct Reference

### Public Types

- typedef [geometric\\_distribution](#)  
<\_IntType> **distribution\_type**

### Public Member Functions

- **param\_type** (double \_\_p=0.5)
- double **p** () const

### Friends

- class **geometric\_distribution**<\_IntType>
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.715.1 Detailed Description

template<typename \_IntType = int> struct std::geometric\_distribution<\_IntType>::param\_type

Parameter type.

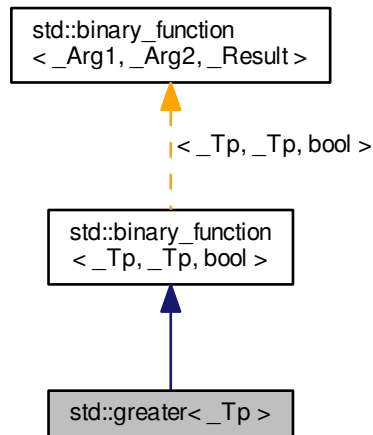
Definition at line 4016 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.716 std::greater&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::greater< \_Tp >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.716.1 Detailed Description

```
template<typename _Tp>struct std::greater< _Tp >
```

One of the [comparison functors](#).

Definition at line 222 of file `stl_function.h`.

## 4.716.2 Member Typedef Documentation

4.716.2.1 typedef `_Tp` `std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.716.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.716.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

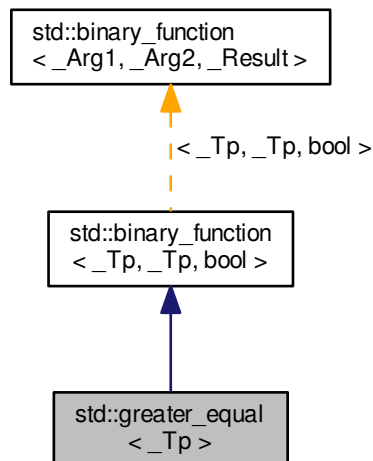
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.717 std::greater\_equal<\_Tp> Struct Template Reference

Inheritance diagram for `std::greater_equal<_Tp>`:



### Public Types

- `typedef _Tp` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.717.1 Detailed Description

template<typename \_Tp>struct std::greater\_equal< \_Tp >

One of the [comparison functors](#).

Definition at line 240 of file stl\_function.h.

## 4.717.2 Member Typedef Documentation

4.717.2.1 typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.717.2.2 typedef bool std::binary\_function< \_Tp, \_Tp, bool >::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

4.717.2.3 typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.718 std::gslice Class Reference

## Public Member Functions

- [gslice](#) ()
- [gslice](#) (size\_t \_\_o, const [valarray](#)< size\_t > &\_\_l, const [valarray](#)< size\_t > &\_\_s)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- size\_t [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

## Friends

- template<typename \_Tp >  
class [valarray](#)

## 4.718.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 64 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

## 4.719 `std::gslice_array< _Tp >` Class Template Reference

### Public Types

- `typedef _Tp value_type`

### Public Member Functions

- [gslice\\_array](#) (const [gslice\\_array](#) &)
- void [operator%>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator%>=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator&gt;=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator&gt;=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator\\*>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator\\*>=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator+>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator+>=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator->=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator->=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator/=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator/=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator<<=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator<<=](#) (const \_Expr< \_Dom, \_Tp > &) const
- [gslice\\_array](#) & [operator=](#) (const [gslice\\_array](#) &)
- void [operator=](#) (const [valarray](#)< \_Tp > &) const
- void [operator=](#) (const \_Tp &) const
- template<class \_Dom >  
void [operator=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator>>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator>>=](#) (const \_Expr< \_Dom, \_Tp > &) const

- void [operator^](#)=(const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator^**=(const \_Expr< \_Dom, \_Tp > &) const
- void [operator|](#)=(const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator|**=(const \_Expr< \_Dom, \_Tp > &) const

#### Friends

- class [valarray](#)< \_Tp >

#### 4.719.1 Detailed Description

template<typename \_Tp>class std::gslice\_array< \_Tp >

Reference to multi-dimensional subset of an array.

A `gslice_array` is a reference to the actual elements of an array specified by a `gslice`. The way to get a `gslice_array` is to call `operator[](gslice)` on a `valarray`. The returned `gslice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `gslice_array` refers to.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

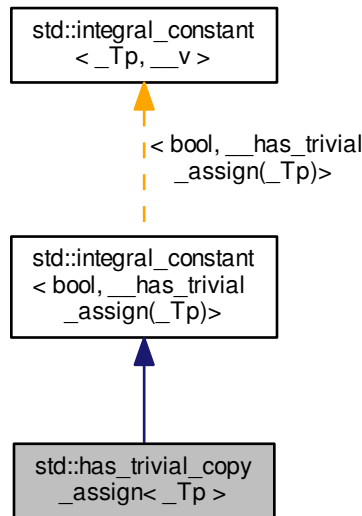
Definition at line 60 of file `gslice_array.h`.

The documentation for this class was generated from the following file:

- [gslice\\_array.h](#)

## 4.720 std::has\_trivial\_copy\_assign&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::has\_trivial\_copy\_assign< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `bool` **value**

## 4.720.1 Detailed Description

```
template<typename _Tp>struct std::has_trivial_copy_assign< _Tp >
```

has\_trivial\_copy\_assign (temporary legacy)

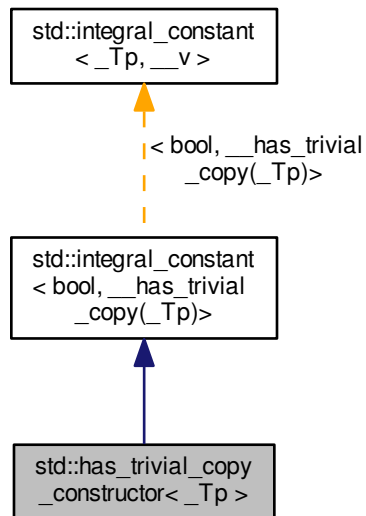
Definition at line 1223 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.721 std::has\_trivial\_copy\_constructor< \_Tp > Struct Template Reference

Inheritance diagram for std::has\_trivial\_copy\_constructor< \_Tp >:



### Public Types

- typedef [integral\\_constant](#) `< bool, __v >` **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 4.721.1 Detailed Description

```
template<typename _Tp>struct std::has_trivial_copy_constructor< _Tp >
```

has\_trivial\_copy\_constructor (temporary legacy)

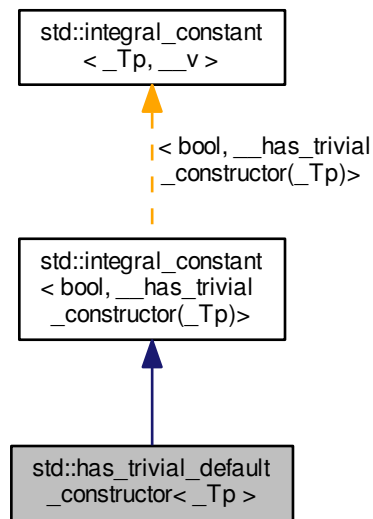
Definition at line 1217 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.722 std::has\_trivial\_default\_constructor< \_Tp > Struct Template Reference

Inheritance diagram for std::has\_trivial\_default\_constructor< \_Tp >:



### Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool `value_type`

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

## 4.722.1 Detailed Description

template<typename \_Tp>struct std::has\_trivial\_default\_constructor< \_Tp >

has\_trivial\_default\_constructor (temporary legacy)

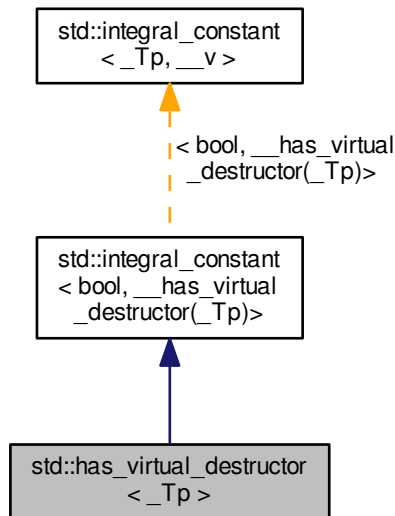
Definition at line 1211 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.723 std::has\_virtual\_destructor&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::has\_virtual\_destructor< \_Tp >:



## Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.723.1 Detailed Description

```
template<typename _Tp>struct std::has_virtual_destructor< _Tp >
```

has\_virtual\_destructor

Definition at line 1229 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.724 `std::hash< __debug::bitset< _Nb > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const `__debug::bitset< _Nb > &__b`) const noexcept

## 4.724.1 Detailed Description

```
template<size_t _Nb>struct std::hash< __debug::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 415 of file `debug/bitset`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

4.725 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const `__debug::vector< bool, _Alloc > &__b`) const noexcept

## 4.725.1 Detailed Description

`template<typename _Alloc> struct std::hash< __debug::vector< bool, _Alloc > >`

`std::hash` specialization for `vector<bool>`.

Definition at line 643 of file `debug/vector`.

The documentation for this struct was generated from the following file:

- [debug/vector](#)

4.726 `std::hash< __gnu_cxx::__u16vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const `__gnu_cxx::__u16vstring &__s`) const noexcept

## 4.726.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::__u16vstring >`

`std::hash` specialization for `__u16vstring`.

Definition at line 2826 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.727 `std::hash< __gnu_cxx::__u32vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_u32vstring](#) &\_\_s) const noexcept

## 4.727.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::__u32vstring >`

`std::hash` specialization for `__u32vstring`.

Definition at line 2837 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.728 `std::hash< __gnu_cxx::__vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_vstring](#) &\_\_s) const noexcept

## 4.728.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::__vstring >`

`std::hash` specialization for `__vstring`.

Definition at line 2802 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.729 `std::hash< __gnu_cxx::__wvstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::\\_\\_wvstring](#) &\_\_s) const noexcept

## 4.729.1 Detailed Description

template<>struct std::hash< \_\_gnu\_cxx::\_\_wvstring >

std::hash specialization for \_\_wvstring.

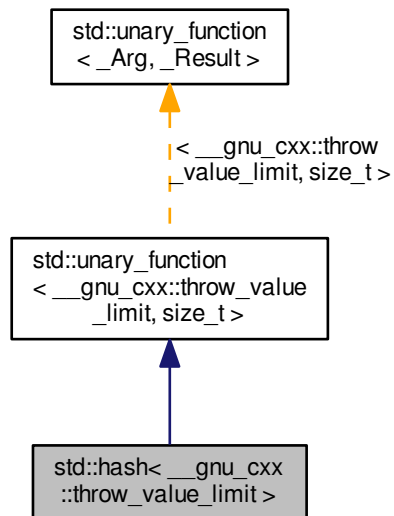
Definition at line 2813 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

## 4.730 std::hash&lt; \_\_gnu\_cxx::throw\_value\_limit &gt; Struct Template Reference

Inheritance diagram for std::hash< \_\_gnu\_cxx::throw\_value\_limit >:



## Public Types

- typedef [\\_\\_gnu\\_cxx::throw\\_value\\_limit argument\\_type](#)
- typedef `size_t` [result\\_type](#)

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#) &\_\_val) const

### 4.730.1 Detailed Description

`template<> struct std::hash<__gnu_cxx::throw_value_limit>`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 788 of file `throw_allocator.h`.

### 4.730.2 Member Typedef Documentation

4.730.2.1 `typedef __gnu_cxx::throw_value_limit std::unary_function<__gnu_cxx::throw_value_limit, size_t>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.730.2.2 `typedef size_t std::unary_function<__gnu_cxx::throw_value_limit, size_t>::result_type` [inherited]

`result_type` is the return type

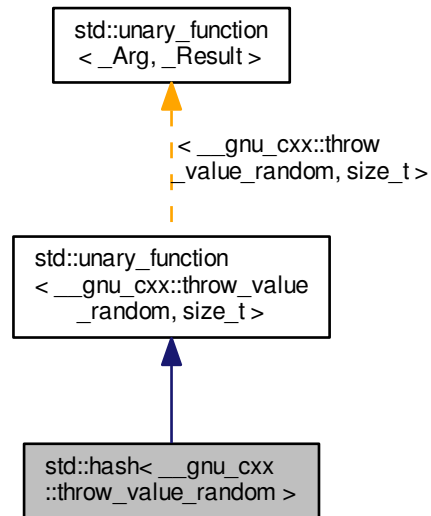
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.731 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



## Public Types

- typedef `__gnu_cxx::throw_value_random argument_type`
- typedef `size_t result_type`

## Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random &__val`) const

## 4.731.1 Detailed Description

template<> struct `std::hash< __gnu_cxx::throw_value_random >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 802 of file `throw_allocator.h`.

## 4.731.2 Member Typedef Documentation

4.731.2.1 `typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random , size_t >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.731.2.2 `typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.732 `std::hash< __profile::bitset< _Nb > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

### Public Member Functions

- `size_t operator() (const __profile::bitset< _Nb > &__b) const` noexcept

### 4.732.1 Detailed Description

`template<size_t _Nb>struct std::hash< __profile::bitset< _Nb > >`

`std::hash` specialization for `bitset`.

Definition at line 371 of file `profile/bitset`.

The documentation for this struct was generated from the following file:

- [profile/bitset](#)

## 4.733 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (const `__profile::vector< bool, _Alloc >` &\_\_b) const noexcept

## 4.733.1 Detailed Description

```
template<typename _Alloc>struct std::hash< __profile::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 532 of file `profile/vector`.

The documentation for this struct was generated from the following file:

- [profile/vector](#)

4.734 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const `__shared_ptr< _Tp, _Lp >` &\_\_s) const noexcept

## 4.734.1 Detailed Description

```
template<typename _Tp, _Lock_policy _Lp>struct std::hash< __shared_ptr< _Tp, _Lp > >
```

`std::hash` specialization for `__shared_ptr`.

Definition at line 1442 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

4.735 `std::hash< _Tp * >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator() (Tp *__p) const` noexcept

##### 4.735.1 Detailed Description

```
template<typename Tp>struct std::hash< Tp * >
```

Partial specializations for pointer types.

Definition at line 62 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.736 std::hash< bool > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- `size_t operator() (bool __val) const` noexcept

##### 4.736.1 Detailed Description

```
template<>struct std::hash< bool >
```

Explicit specialization for bool.

Definition at line 80 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.737 std::hash< char > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- `size_t operator()` (`char __val`) `const noexcept`

## 4.737.1 Detailed Description

`template<> struct std::hash< char >`

Explicit specialization for `char`.

Definition at line 83 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.738 `std::hash< char16_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (`char16_t __val`) `const noexcept`

## 4.738.1 Detailed Description

`template<> struct std::hash< char16_t >`

Explicit specialization for `char16_t`.

Definition at line 95 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.739 `std::hash< char32_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (`char32_t __val`) `const noexcept`

## 4.739.1 Detailed Description

```
template<> struct std::hash< char32_t >
```

Explicit specialization for `char32_t`.

Definition at line 98 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.740 `std::hash< double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (`double __val`) `const noexcept`

## 4.740.1 Detailed Description

```
template<> struct std::hash< double >
```

Specialization for `double`.

Definition at line 176 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.741 `std::hash< error_code >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `size_t operator()` (const [error\\_code](#) &\_\_e) const noexcept

## 4.741.1 Detailed Description

`template<> struct std::hash< error_code >`

`std::hash` specialization for `error_code`.

Definition at line 360 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

4.742 `std::hash< float >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (float \_\_val) const noexcept

## 4.742.1 Detailed Description

`template<> struct std::hash< float >`

Specialization for `float`.

Definition at line 164 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.743 `std::hash< int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (int \_\_val) const noexcept

## 4.743.1 Detailed Description

`template<> struct std::hash< int >`

Explicit specialization for int.

Definition at line 104 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.744 `std::hash< long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (long \_\_val) const noexcept

## 4.744.1 Detailed Description

`template<> struct std::hash< long >`

Explicit specialization for long.

Definition at line 107 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.745 `std::hash< long double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (long double \_\_val) const noexcept

##### 4.745.1 Detailed Description

template<>struct std::hash< long double >

Specialization for long double.

Definition at line 188 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.746 std::hash< long long > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- `size_t operator()` (long long \_\_val) const noexcept

##### 4.746.1 Detailed Description

template<>struct std::hash< long long >

Explicit specialization for long long.

Definition at line 110 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.747 std::hash< shared\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- `size_t operator()` (const [shared\\_ptr](#)< \_Tp > &\_\_s) const noexcept

## 4.747.1 Detailed Description

```
template<typename _Tp>struct std::hash< shared_ptr< _Tp > >
```

`std::hash` specialization for `shared_ptr`.

Definition at line 619 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

4.748 `std::hash< short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (short \_\_val) const noexcept

## 4.748.1 Detailed Description

```
template<>struct std::hash< short >
```

Explicit specialization for `short`.

Definition at line 101 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.749 `std::hash< signed char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (signed char \_\_val) const noexcept

## 4.749.1 Detailed Description

`template<> struct std::hash< signed char >`

Explicit specialization for signed char.

Definition at line 86 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.750 `std::hash< string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [string](#) &\_\_s) const noexcept

## 4.750.1 Detailed Description

`template<> struct std::hash< string >`

`std::hash` specialization for `string`.

Definition at line 3046 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

4.751 `std::hash< thread::id >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [thread::id](#) &\_\_id) const noexcept

## 4.751.1 Detailed Description

`template<>struct std::hash< thread::id >`

`std::hash` specialization for `thread::id`.

Definition at line 222 of file `thread`.

The documentation for this struct was generated from the following file:

- [thread](#)

4.752 `std::hash< type_index >` Struct Template Reference

## Public Types

- typedef [type\\_index](#) **argument\_type**
- typedef `size_t` **result\_type**

## Public Member Functions

- `size_t operator()` (const [type\\_index](#) &\_\_ti) const noexcept

## 4.752.1 Detailed Description

`template<>struct std::hash< type_index >`

`std::hash` specialization for `type_index`.

Definition at line 97 of file `typeindex`.

The documentation for this struct was generated from the following file:

- [typeindex](#)

4.753 `std::hash< u16string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [u16string](#) &\_\_s) const noexcept

## 4.753.1 Detailed Description

`template<> struct std::hash< u16string >`

`std::hash` specialization for `u16string`.

Definition at line 3079 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

4.754 `std::hash< u32string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [u32string](#) &\_\_s) const noexcept

## 4.754.1 Detailed Description

`template<> struct std::hash< u32string >`

`std::hash` specialization for `u32string`.

Definition at line 3094 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

4.755 `std::hash< unique_ptr< _Tp, _Dp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [unique\\_ptr](#)< `_Tp`, `_Dp` > &\_\_u) const noexcept

## 4.755.1 Detailed Description

```
template<typename _Tp, typename _Dp> struct std::hash< unique_ptr< _Tp, _Dp > >
```

std::hash specialization for unique\_ptr.

Definition at line 599 of file unique\_ptr.h.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 4.756 std::hash&lt; unsigned char &gt; Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- size\_t **operator()** (unsigned char \_\_val) const noexcept

## 4.756.1 Detailed Description

```
template<> struct std::hash< unsigned char >
```

Explicit specialization for unsigned char.

Definition at line 89 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.757 std::hash&lt; unsigned int &gt; Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- size\_t **operator()** (unsigned int \_\_val) const noexcept

## 4.757.1 Detailed Description

template<>struct std::hash< unsigned int >

Explicit specialization for unsigned int.

Definition at line 116 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.758 std::hash&lt; unsigned long &gt; Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- size\_t **operator()** (unsigned long \_\_val) const noexcept

## 4.758.1 Detailed Description

template<>struct std::hash< unsigned long >

Explicit specialization for unsigned long.

Definition at line 119 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.759 std::hash&lt; unsigned long long &gt; Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- size\_t **operator()** (unsigned long long \_\_val) const noexcept

## 4.759.1 Detailed Description

`template<> struct std::hash< unsigned long long >`

Explicit specialization for unsigned long long.

Definition at line 122 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.760 `std::hash< unsigned short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator() (unsigned short __val) const` noexcept

## 4.760.1 Detailed Description

`template<> struct std::hash< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 113 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.761 `std::hash< wchar_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator() (wchar_t __val) const` noexcept

## 4.761.1 Detailed Description

template<>struct std::hash< wchar\_t >

Explicit specialization for wchar\_t.

Definition at line 92 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.762 std::hash&lt; wstring &gt; Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- size\_t **operator()** (const [wstring](#) &\_\_s) const noexcept

## 4.762.1 Detailed Description

template<>struct std::hash< wstring >

std::hash specialization for wstring.

Definition at line 3061 of file basic\_string.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 4.763 std::hash&lt;::bitset&lt; \_Nb &gt; &gt; Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

## Public Member Functions

- size\_t **operator()** (const ::bitset< \_Nb > &\_\_b) const noexcept

## 4.763.1 Detailed Description

```
template<size_t _Nb>struct std::hash<::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 1552 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

4.764 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t` **operator()** (const `::vector< bool, _Alloc > &`) const noexcept

## 4.764.1 Detailed Description

```
template<typename _Alloc>struct std::hash<::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 1143 of file `stl_bvector.h`.

The documentation for this struct was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

4.765 `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >` Class Template Reference

## Public Types

- typedef `_UIntType` **result\_type**

## Public Member Functions

- [independent\\_bits\\_engine](#) ()
- [independent\\_bits\\_engine](#) (const `_RandomNumberEngine &__rng`)
- [independent\\_bits\\_engine](#) (`_RandomNumberEngine &&__rng`)
- [independent\\_bits\\_engine](#) (**result\_type** \_\_s)

- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`  
`independent_bits_engine` (`_Sseq` & `__q`)
- `const _RandomNumberEngine & base` () `const` `noexcept`
- `void discard` (`unsigned long long __z`)
- `result_type operator` ()
- `void seed` ()
- `void seed` (`result_type __s`)
- `template<typename _Sseq>`  
`void seed` (`_Sseq` & `__q`)

#### Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

#### Friends

- `bool operator==` (`const independent_bits_engine` & `__lhs`, `const independent_bits_engine` & `__rhs`)
- `template<typename _CharT, typename _Traits>`  
`std::basic_istream` < `_CharT`,  
`_Traits` > & `operator>>` (`std::basic_istream` < `_CharT`, `_Traits` > & `__is`, `std::independent_bits_engine` < `_RandomNumberEngine`, `__w`, `_UIntType` > & `__x`)

#### 4.765.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __w, typename _UIntType> class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1074 of file `random.h`.

#### 4.765.2 Member Typedef Documentation

4.765.2.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> typedef _UIntType`  
`std::independent_bits_engine` < `_RandomNumberEngine`, `__w`, `_UIntType` > :: `result_type`

The type of the generated random value.

Definition at line 1077 of file `random.h`.

#### 4.765.3 Constructor & Destructor Documentation

4.765.3.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine` <  
`_RandomNumberEngine`, `__w`, `_UIntType` > :: `independent_bits_engine` ( ) `[inline]`

Constructs a default `independent_bits_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1090 of file `random.h`.

```
4.765.3.2 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<
 _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (const _RandomNumberEngine & __rng)
 [inline], [explicit]
```

Copy constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1100 of file random.h.

```
4.765.3.3 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<
 _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (_RandomNumberEngine && __rng)
 [inline], [explicit]
```

Move constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1110 of file random.h.

```
4.765.3.4 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<
 _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (result_type __s) [inline],
 [explicit]
```

Seed constructs a independent\_bits\_engine engine.

Constructs the underlying generator engine seeded with `__s`.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

Definition at line 1120 of file random.h.

```
4.765.3.5 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq, typename
 = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq,
 _RandomNumberEngine>::value> ::type> std::independent_bits_engine< _RandomNumberEngine, __w,
 _UIntType >::independent_bits_engine (_Sseq & __q) [inline], [explicit]
```

Generator construct a independent\_bits\_engine engine.

#### Parameters

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

Definition at line 1133 of file random.h.

## 4.765.4 Member Function Documentation

```
4.765.4.1 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> const _RandomNumberEngine&
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::base () const [inline],
[noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1168 of file random.h.

Referenced by std::operator<<().

```
4.765.4.2 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent-
_bits_engine<_RandomNumberEngine, __w, _UIntType >::discard (unsigned long long __z)
[inline]
```

Discard a sequence of random numbers.

Definition at line 1189 of file random.h.

```
4.765.4.3 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::max () [inline],[static]
```

Gets the maximum value in the generated random number range.

Definition at line 1182 of file random.h.

```
4.765.4.4 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::min () [inline],[static]
```

Gets the minimum value in the generated random number range.

Definition at line 1175 of file random.h.

```
4.765.4.5 template<typename _RandomNumberEngine , size_t __w, typename _UIntType > independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<_RandomNumberEngine,
__w, _UIntType >::operator() ()
```

Gets the next value in the generated random number sequence.

Definition at line 745 of file bits/random.tcc.

References std::\_\_lg(), std::numeric\_limits<\_Tp >::max(), and std::numeric\_limits<\_Tp >::min().

```
4.765.4.6 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::seed () [inline]
```

Reseeds the independent\_bits\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1142 of file random.h.

```
4.765.4.7 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std-
::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::seed (result_type __s)
[inline]
```

Reseeds the independent\_bits\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1150 of file random.h.

```
4.765.4.8 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq > void
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::seed (_Sseq & __q) [inline]
```

Reseeds the independent\_bits\_engine object with the given seed sequence.

## Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__q</code> | A seed generator function. |
|------------------|----------------------------|

Definition at line 1160 of file random.h.

## 4.765.5 Friends And Related Function Documentation

4.765.5.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator==  
( const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs, const  
independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs ) [friend]`

Compares two independent\_bits\_engine random number generator objects of the same type for equality.

## Parameters

|                    |                                                                 |
|--------------------|-----------------------------------------------------------------|
| <code>__lhs</code> | A independent_bits_engine random number generator object.       |
| <code>__rhs</code> | Another independent_bits_engine random number generator object. |

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1214 of file random.h.

4.765.5.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT, typename  
_Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x ) [friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

## Parameters

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <code>__is</code> | An input stream.                                          |
| <code>__x</code>  | A independent_bits_engine random number generator engine. |

## Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1232 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.766 std::indirect\_array&lt;\_Tp&gt; Class Template Reference

## Public Types

- `typedef _Tp value_type`

## Public Member Functions

- [indirect\\_array](#) (const [indirect\\_array](#) &)
- void [operator%=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator%=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator&=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator&=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator\\*=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator\\*=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator+=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator+=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator-=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator-=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator/=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator/=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator<<=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator<<=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- [indirect\\_array](#) & [operator=](#) (const [indirect\\_array](#) &)
- void [operator=](#) (const [valarray](#)<\_Tp> &) const
- void [operator=](#) (const \_Tp &) const
- template<class \_Dom >  
void [operator=](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator>>=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator>>=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator^=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator^=>](#) (const \_Expr<\_Dom, \_Tp> &) const
- void [operator|=>](#) (const [valarray](#)<\_Tp> &) const
- template<class \_Dom >  
void [operator|=>](#) (const \_Expr<\_Dom, \_Tp> &) const

## Friends

- class [gslice\\_array](#)<\_Tp>
- class [valarray](#)<\_Tp>

## 4.766.1 Detailed Description

template<class \_Tp>class std::indirect\_array<\_Tp>

Reference to arbitrary subset of an array.

An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

Definition at line 62 of file `indirect_array.h`.

The documentation for this class was generated from the following file:

- [indirect\\_array.h](#)

## 4.767 std::initializer\_list< \_E > Class Template Reference

### Public Types

- typedef const \_E \* **const\_iterator**
- typedef const \_E & **const\_reference**
- typedef const \_E \* **iterator**
- typedef const \_E & **reference**
- typedef size\_t **size\_type**
- typedef \_E **value\_type**

### Public Member Functions

- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **size** () const noexcept

#### 4.767.1 Detailed Description

```
template<class _E>class std::initializer_list< _E >
```

`initializer_list`

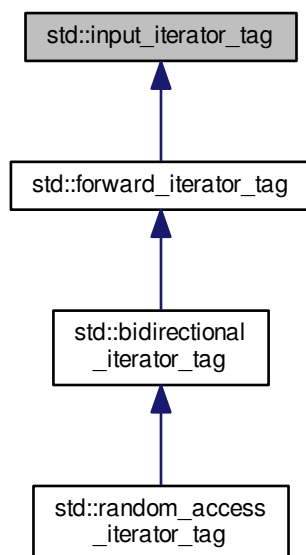
Definition at line 47 of file `initializer_list`.

The documentation for this class was generated from the following file:

- [initializer\\_list](#)

## 4.768 std::input\_iterator\_tag Struct Reference

Inheritance diagram for std::input\_iterator\_tag:



### 4.768.1 Detailed Description

Marking input iterators.

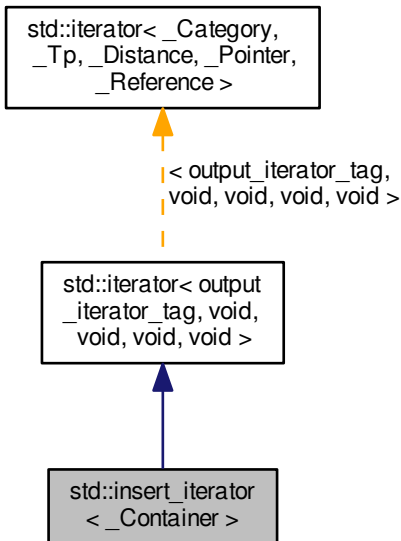
Definition at line 89 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.769 std::insert\_iterator&lt; \_Container &gt; Class Template Reference

Inheritance diagram for std::insert\_iterator< \_Container >:



## Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

## Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator` & `operator*` ()
- `insert_iterator` & `operator++` ()
- `insert_iterator` & `operator++` (int)
- `insert_iterator` & `operator=` (const `typename _Container::value_type &__value`)
- `insert_iterator` & `operator=` (`typename _Container::value_type &&__value`)

## Protected Attributes

- `_Container * container`
- `_Container::iterator iter`

#### 4.769.1 Detailed Description

template<typename \_Container>class std::insert\_iterator< \_Container >

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 587 of file stl\_iterator.h.

#### 4.769.2 Member Typedef Documentation

4.769.2.1 template<typename \_Container> typedef \_Container std::insert\_iterator< \_Container >::container\_type

A nested typedef for the type of whatever container you used.

Definition at line 596 of file stl\_iterator.h.

4.769.2.2 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference\_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

4.769.2.3 typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void , void , void , void >::iterator\_category [inherited]

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

4.769.2.4 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::pointer [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

4.769.2.5 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::reference [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

4.769.2.6 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::value\_type [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

#### 4.769.3 Constructor & Destructor Documentation

4.769.3.1 `template<typename _Container> std::insert_iterator<_Container>::insert_iterator ( _Container & __x, typename _Container::iterator __i ) [inline]`

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 602 of file `stl_iterator.h`.

#### 4.769.4 Member Function Documentation

4.769.4.1 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator* ( ) [inline]`

Simply returns `*this`.

Definition at line 656 of file `stl_iterator.h`.

4.769.4.2 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator++ ( ) [inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 661 of file `stl_iterator.h`.

4.769.4.3 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator++ ( int ) [inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 666 of file `stl_iterator.h`.

4.769.4.4 `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator= ( const typename _Container::value_type & __value ) [inline]`

#### Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

Definition at line 638 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

4.770 `std::integral_constant< _Tp, __v >` Struct Template Reference

Inherited by `std::__and<>`, `std::__is_base_to_derived_ref< _From, _To, false >`, `std::__is_copy_assignable_impl< _Tp, true >`, `std::__is_copy_constructible_impl< _Tp, true >`, `std::__is_destructible_safe< _Tp, false, true >`, `std::__is_destructible_safe< _Tp, true, false >`, `std::__is_empty_non_tuple< tuple< _El0, _El...> >`, `std::__is_fast_hash< _Hash >`, `std::__is_fast_hash< hash< long double > >`, `std::__is_fast_hash< hash< string > >`, `std::__is_fast_hash< hash< u16string > >`, `std::__is_fast_hash< hash< u32string > >`, `std::__is_fast_hash< hash< wstring > >`, `std::__is_floating_point_helper< typename >`, `std::__is_floating_point_helper< double >`, `std::__is_floating_point_helper< float >`, `std::__is_floating_point_helper< long double >`, `std::__is_integral_helper< typename >`, `std::__is_integral_helper< bool >`, `std::__is_integral_helper< char >`, `std::__is_integral_helper< char16_t >`, `std::__is_integral_helper< char32_t >`, `std::__is_integral_helper< int >`, `std::__is_integral_helper< long >`, `std::__is_integral_helper< long long >`, `std::__is_integral_helper< short >`, `std::__is_integral_helper< signed char >`, `std::__is_integral_helper< unsigned char >`, `std::__is_integral_helper< unsigned int >`, `std::__is_integral_helper< unsigned long >`, `std::__is_integral_helper< unsigned long long >`, `std::__is_integral_helper< unsigned short >`, `std::__is_integral_helper< wchar_t >`, `std::__is_lvalue_to_rvalue_ref< _From, _To, false >`, `std::__is_member_function_pointer_helper< typename >`, `std::__is_member_object_pointer_helper< typename >`, `std::__is_member_pointer_helper< _Tp >`, `std::__is_member_pointer_helper< _Tp Cp::* >`, `std::__is_move_assignable_impl< _Tp, true >`, `std::__is_move_constructible_impl< _Tp, true >`, `std::__is_nothrow_copy_constructible_impl< _Tp, true >`, `std::__is_nothrow_move_constructible_impl< _Tp, true >`, `std::__is_nt_copy_assignable_impl< _Tp, true >`, `std::__is_nt_destructible_safe< _Tp, false, true >`, `std::__is_nt_destructible_safe< _Tp, true, false >`, `std::__is_nt_move_assignable_impl< _Tp, true >`, `std::__is_nullptr_t_helper< typename >`, `std::__is_nullptr_t_helper< std::nullptr_t >`, `std::__is_pointer_helper< typename >`, `std::__is_pointer_helper< _Tp * >`, `std::__is_signed_helper< _Tp, bool, bool >`, `std::__is_signed_helper< _Tp, false, true >`, `std::__is_tuple_like< _Tp >`, `std::__is_tuple_like_impl< typename >`, `std::__is_tuple_like_impl< array< _Tp, _Nm > >`, `std::__is_tuple_like_impl< pair< _T1, _T2 > >`, `std::__is_tuple_like_impl< tuple< _Tps...> >`, `std::__is_void_helper< typename >`, `std::__is_void_helper< void >`, `std::__or<>`, `std::__ratio_less_impl< _R1, _R2, bool, bool >`, `std::__uses_allocator_helper< _Tp, _Alloc, bool >`, `std::AllConvertible< _From, _To, bool >`, `std::chrono::__is_duration< _Tp >`, `std::chrono::__is_duration< duration< _Rep, _Period > >`, `std::chrono::__is_ratio< T >`, `std::chrono::__is_ratio< ratio< _Num, _Den > >`, `std::function< _Res(_ArgTypes...)>::CheckResult< _CallRes, void >`, `std::is_array< typename >`, `std::is_array< _Tp[] >`, `std::is_array< _Tp[_Size] >`, `std::is_bind_expression< _Tp >`, `std::is_bind_expression< _Bind< _Signature > >`, `std::is_bind_expression< _Bind_result< _Result, _Signature > >`, `std::is_bind_expression< const _Bind< _Signature > >`, `std::is_bind_expression< const _Bind_result< _Result, _Signature > >`, `std::is_bind_expression< const volatile _Bind< _Signature > >`, `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`, `std::is_bind_expression< volatile _Bind< _Signature > >`, `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`, `std::is_const< typename >`, `std::is_const< _Tp const >`, `std::is_error_code_enum< _Tp >`, `std::is_error_code_enum< future_errc >`, `std::is_error_condition_enum< _Tp >`, `std::is_error_condition_enum< errc >`, `std::is_function< typename >`, `std::is_function< _Res(_ArgTypes...) const >`, `std::is_function< _Res(_ArgTypes...) const volatile >`, `std::is_function< _Res(_ArgTypes...) volatile >`, `std::is_function< _Res(_ArgTypes.....) const >`, `std::is_function< _Res(_ArgTypes.....) const volatile >`, `std::is_function< _Res(_ArgTypes.....) volatile >`, `std::is_function< _Res(_ArgTypes.....)>`, `std::is_lvalue_reference< typename >`, `std::is_lvalue_reference< _Tp & >`, `std::is_object< _Tp >`, `std::is_rvalue_reference< typename >`, `std::is_rvalue_reference< _Tp && >`, `std::is_same< typename, typename >`, `std::is_same< _Tp, _Tp >`, `std::is_volatile< typename >`, `std::is_volatile< _Tp volatile >`, `std::uses_allocator< priority_queue< _Tp, _Sequence, _Compare >, _Alloc >`, `std::uses_allocator< queue< _Tp, _Seq >, _Alloc >`, `std::uses_allocator< stack< _Tp, _Seq >, _Alloc >`, and `std::uses_allocator< tuple< _Types...>, _Alloc >`.

## Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.770.1 Detailed Description

```
template<typename _Tp, _Tp __v>struct std::integral_constant< _Tp, __v >
```

`integral_constant`

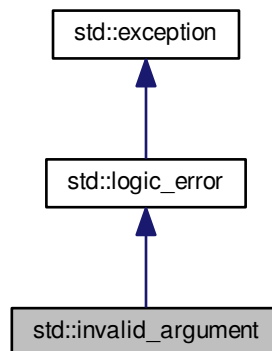
Definition at line 57 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.771 `std::invalid_argument` Class Reference

Inheritance diagram for `std::invalid_argument`:



## Public Member Functions

- **invalid\_argument** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

#### 4.771.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 82 of file stdexcept.

## 4.771.2 Member Function Documentation

#### 4.771.2.1 virtual const char\* std::logic\_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

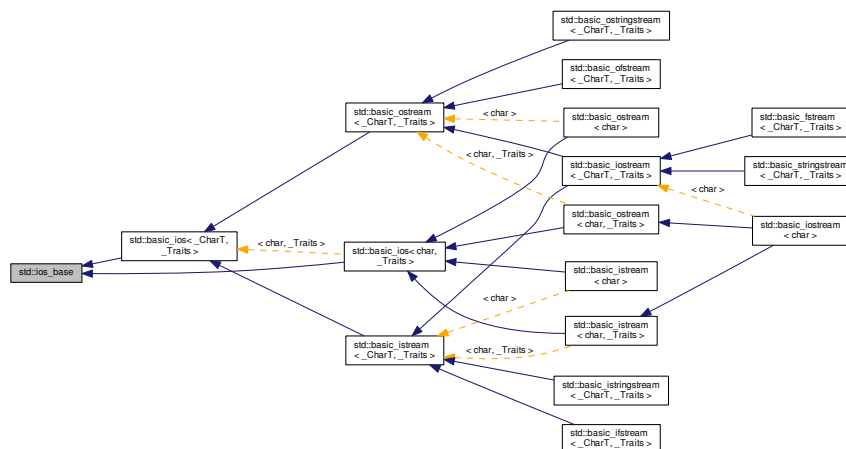
Reimplemented in `std::future` error.

The documentation for this class was generated from the following file:

- `stdexcept`

## 4.772 std::ios\_base Class Reference

Inheritance diagram for `std::ios_base`:



## Classes

- class failure

These are thrown to indicate problems with io.

## Public Types

- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* `event_callback`)(`event` \_\_e, `ios_base` &\_\_b, int \_\_i)

- typedef `_ios_Fmtflags` [fmtflags](#)
- typedef int `io_state`
- typedef `_ios_istate` [istate](#)
- typedef int `open_mode`
- typedef `_ios_Openmode` [openmode](#)
- typedef int `seek_dir`
- typedef `_ios_Seekdir` [seekdir](#)
- typedef [std::streamoff](#) `streamoff`
- typedef [std::streampos](#) `streampos`

#### Public Member Functions

- virtual [~ios\\_base](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- [locale](#) [imbue](#) (const [locale](#) & \_\_loc) throw ()
- long & [iword](#) (int \_\_ix)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [istate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [istate](#) [eofbit](#)
- static const [istate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)

- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

#### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

#### Protected Member Functions

- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()

#### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- [iostate](#) [\\_M\\_exception](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- [streamsize](#) [\\_M\\_precision](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

#### 4.772.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 199 of file `ios_base.h`.

## 4.772.2 Member Typedef Documentation

## 4.772.2.1 typedef void(\* std::ios\_base::event\_callback)(event \_\_e, ios\_base &amp;\_\_b, int \_\_i)

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the ios_base object.                      |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 436 of file ios\_base.h.

## 4.772.2.2 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios\_base.h.

#### 4.772.2.3 typedef \_Ios\_Iostate std::ios\_base::iostate

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios\_base.h.

#### 4.772.2.4 typedef \_Ios\_Openmode std::ios\_base::openmode

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios\_base.h.

#### 4.772.2.5 typedef \_Ios\_Seekdir std::ios\_base::seekdir

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios\_base.h.

### 4.772.3 Member Enumeration Documentation

#### 4.772.3.1 enum std::ios\_base::event

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file ios\_base.h.

## 4.772.4 Constructor &amp; Destructor Documentation

## 4.772.4.1 virtual std::ios\_base::~~ios\_base ( ) [virtual]

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

## 4.772.5 Member Function Documentation

## 4.772.5.1 const locale&amp; std::ios\_base::M\_getloc ( ) const [inline]

Locale access.

## Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

## 4.772.5.2 fmtflags std::ios\_base::flags ( ) const [inline]

Access to format flags.

## Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

## 4.772.5.3 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline]

Setting new format flags all at once.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

## Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

**4.772.5.4 locale std::ios\_base::getloc ( ) const [inline]**

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outlter>::do_put()`, `std::operator>>()`, and `std::ws()`.

**4.772.5.5 locale std::ios\_base::imbue ( const locale & \_\_loc ) throw ()**

Setting a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Sets the new locale for this stream, and then invokes each callback with `imbue_event`.

Referenced by `std::basic_ios<_CharT, _Traits>::imbue()`.

**4.772.5.6 long& std::ios\_base::iword ( int \_\_ix ) [inline]**

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

**4.772.5.7 streamsize std::ios\_base::precision ( ) const [inline]**

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::operator<<().

**4.772.5.8** streamsize std::ios\_base::precision ( streamsize \_\_prec ) [inline]

Changing flags.

**Parameters**

|        |                          |
|--------|--------------------------|
| __prec | The new precision value. |
|--------|--------------------------|

**Returns**

The previous value of precision().

Definition at line 630 of file ios\_base.h.

**4.772.5.9** void\*& std::ios\_base::pword ( int \_\_ix ) [inline]

Access to void pointer array.

**Parameters**

|      |                       |
|------|-----------------------|
| __ix | Index into the array. |
|------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios\_base.h.

**4.772.5.10** void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index )

Add the callback \_\_fn with parameter \_\_index.

**Parameters**

|         |                                                   |
|---------|---------------------------------------------------|
| __fn    | The function to add.                              |
| __index | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.772.5.11** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline]`

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**4.772.5.12** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline]`

Setting new format flags.

**Parameters**

|                      |                                           |
|----------------------|-------------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                  |
| <code>__mask</code>  | The flags mask for <code>__fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `__fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

**4.772.5.13** `static bool std::ios_base::sync_with_stdio ( bool __sync = true ) [static]`

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

**4.772.5.14** `void std::ios_base::unsetf ( fmtflags __mask ) [inline]`

Clearing format flags.

## Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

#### 4.772.5.15 streamsize std::ios\_base::width ( ) const [inline]

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

#### 4.772.5.16 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline]

Changing flags.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

## Returns

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

#### 4.772.5.17 static int std::ios\_base::xalloc ( ) throw () [static]

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 4.772.6 Member Data Documentation

**4.772.6.1 const fmtflags std::ios\_base::adjustfield** [static]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**4.772.6.2 const openmode std::ios\_base::app** [static]

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

**4.772.6.3 const openmode std::ios\_base::ate** [static]

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**4.772.6.4 const iostate std::ios\_base::badbit** [static]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

**4.772.6.5 const fmtflags std::ios\_base::basefield** [static]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**4.772.6.6 const seekdir std::ios\_base::beg** [static]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**4.772.6.7 const openmode std::ios\_base::binary** [static]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

#### 4.772.6.8 `const fmtflags std::ios_base::boolalpha` [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

#### 4.772.6.9 `const seekdir std::ios_base::cur` [static]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### 4.772.6.10 `const fmtflags std::ios_base::dec` [static]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.772.6.11 `const seekdir std::ios_base::end` [static]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### 4.772.6.12 `const iostate std::ios_base::eofbit` [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

#### 4.772.6.13 `const iostate std::ios_base::failbit` [static]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 4.772.6.14 `const fmtflags std::ios_base::fixed` [static]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file `ios_base.h`.

Referenced by `std::fixed()`.

#### 4.772.6.15 `const fmtflags std::ios_base::floatfield` [static]

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 316 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

#### 4.772.6.16 `const iostate std::ios_base::goodbit` [static]

Indicates all is well.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unset()`.

#### 4.772.6.17 `const fmtflags std::ios_base::hex` [static]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 4.772.6.18 `const openmode std::ios_base::in` [static]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_`

`_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

#### 4.772.6.19 `const fmtflags std::ios_base::internal` `[static]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.772.6.20 `const fmtflags std::ios_base::left` `[static]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, and `std::left()`.

#### 4.772.6.21 `const fmtflags std::ios_base::oct` `[static]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 4.772.6.22 `const openmode std::ios_base::out` `[static]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

#### 4.772.6.23 `const fmtflags std::ios_base::right` `[static]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

#### 4.772.6.24 `const fmtflags std::ios_base::scientific` `[static]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

#### 4.772.6.25 `const fmtflags std::ios_base::showbase` `[static]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**4.772.6.26 const fmtflags std::ios\_base::showpoint** [static]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**4.772.6.27 const fmtflags std::ios\_base::showpos** [static]

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**4.772.6.28 const fmtflags std::ios\_base::skipws** [static]

Skips leading white space before certain input operations.

Definition at line 300 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream<\_CharT, \_Traits>::sentry::sentry(), and std::skipws().

**4.772.6.29 const openmode std::ios\_base::trunc** [static]

Open for input. Default for ofstream.

Definition at line 381 of file ios\_base.h.

**4.772.6.30 const fmtflags std::ios\_base::unitbuf** [static]

Flushes output after each output operation.

Definition at line 303 of file ios\_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic\_ostream<\_CharT, \_Traits>::sentry::~sentry().

**4.772.6.31 const fmtflags std::ios\_base::uppercase** [static]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios\_base.h.

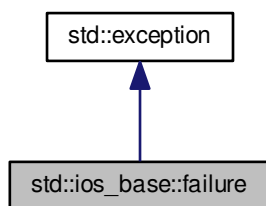
Referenced by std::num\_put<\_CharT, \_Outlter>::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 4.773 `std::ios_base::failure` Class Reference

Inheritance diagram for `std::ios_base::failure`:



### Public Member Functions

- **failure** (const [string](#) &\_\_str) throw ()
- virtual const char \* [what](#) () const throw ()

#### 4.773.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

Definition at line 209 of file `ios_base.h`.

#### 4.773.2 Member Function Documentation

**4.773.2.1** virtual const char\* `std::ios_base::failure::what ( ) const throw ()` [virtual]

Returns a C-style character string describing the general cause of the current error.

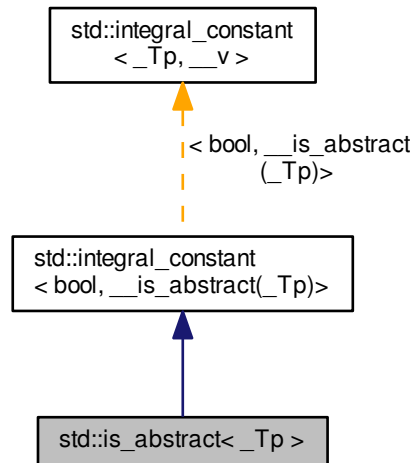
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

4.774 `std::is_abstract<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_abstract<_Tp>`:



## Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `bool` **value**

## 4.774.1 Detailed Description

```
template<typename _Tp> struct std::is_abstract<_Tp>
```

`is_abstract`

Definition at line 548 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

## 4.775 std::is\_arithmetic< \_Tp > Struct Template Reference

Inherits type< is\_integral< \_Tp >, is\_floating\_point< \_Tp > >.

### 4.775.1 Detailed Description

```
template<typename _Tp>struct std::is_arithmetic< _Tp >
```

is\_arithmetic

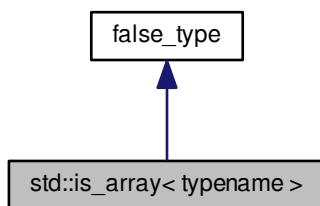
Definition at line 440 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.776 std::is\_array< typename > Struct Template Reference

Inheritance diagram for std::is\_array< typename >:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr \_Tp **value**

## 4.776.1 Detailed Description

```
template<typename>struct std::is_array< typename >
```

is\_array

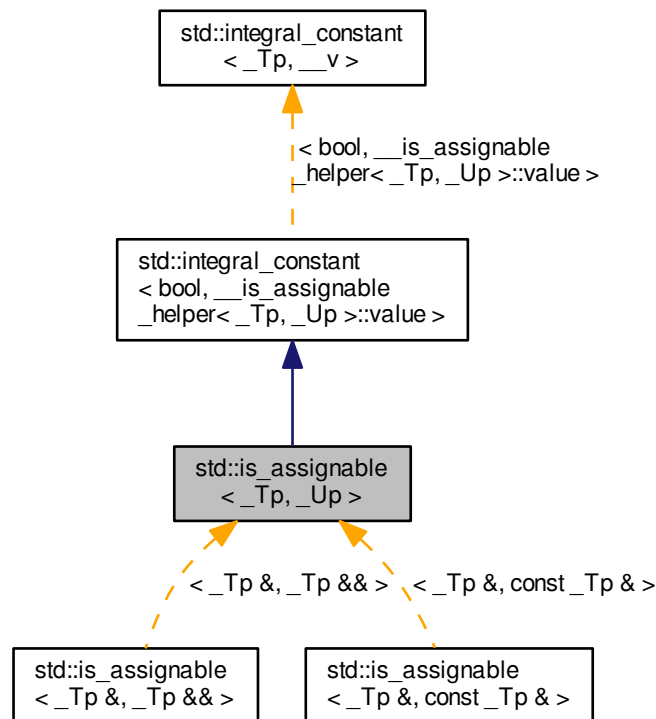
Definition at line 282 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.777 std::is\_assignable&lt; \_Tp, \_Up &gt; Struct Template Reference

Inheritance diagram for std::is\_assignable< \_Tp, \_Up >:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.777.1 Detailed Description

template<typename \_Tp, typename \_Up>struct std::is\_assignable< \_Tp, \_Up >

is\_assignable

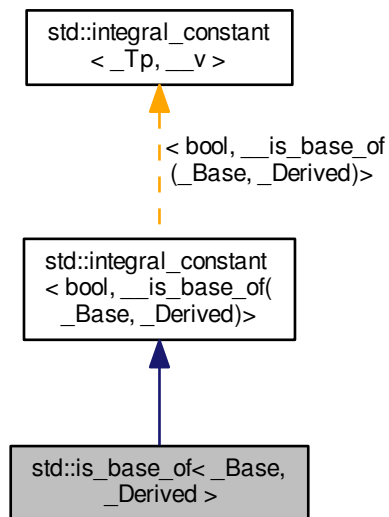
Definition at line 1099 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.778 std::is\_base\_of&lt; \_Base, \_Derived &gt; Struct Template Reference

Inheritance diagram for std::is\_base\_of< \_Base, \_Derived >:



## Public Types

- typedef [integral\\_constant](#)  
`< bool, __v >` **type**

- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** ()

#### Static Public Attributes

- static constexpr bool **value**

#### 4.778.1 Detailed Description

template<typename \_Base, typename \_Derived> struct std::is\_base\_of< \_Base, \_Derived >

is\_base\_of

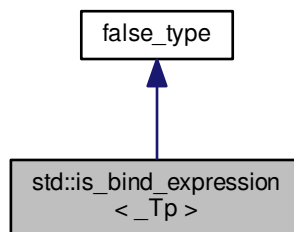
Definition at line 1287 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.779 std::is\_bind\_expression< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_bind\_expression< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.779.1 Detailed Description

```
template<typename _Tp>struct std::is_bind_expression<_Tp>
```

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].

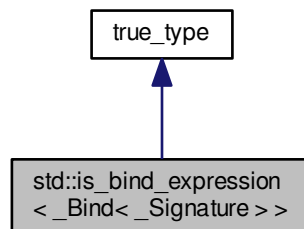
Definition at line 973 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.780 `std::is_bind_expression<_Bind<_Signature>>>` Struct Template Reference

Inheritance diagram for `std::is_bind_expression<_Bind<_Signature>>>`:



## Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.780.1 Detailed Description

```
template<typename _Signature>struct std::is_bind_expression< _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

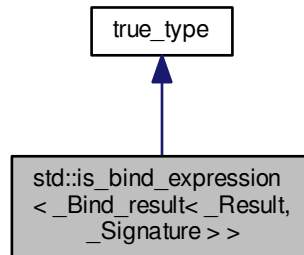
Definition at line 1567 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.781 `std::is_bind_expression< _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.781.1 Detailed Description

```
template<typename _Result, typename _Signature>struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

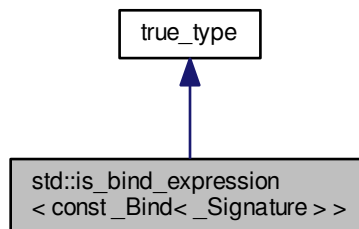
Definition at line 1599 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.782 std::is\_bind\_expression< const \_Bind< \_Signature > > Struct Template Reference

Inheritance diagram for std::is\_bind\_expression< const \_Bind< \_Signature > >:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr \_Tp **value**

### 4.782.1 Detailed Description

```
template<typename _Signature>struct std::is_bind_expression< const _Bind< _Signature > >
```

Class template \_Bind is always a bind expression.

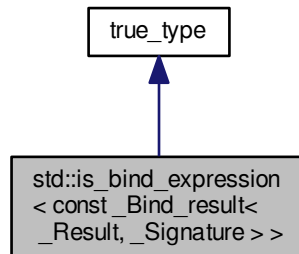
Definition at line 1575 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.783 `std::is_bind_expression< const _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const _Bind_result< _Result, _Signature > >`:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.783.1 Detailed Description

`template<typename _Result, typename _Signature>struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >`

Class template `_Bind_result` is always a bind expression.

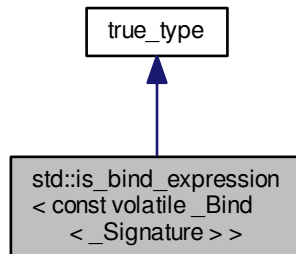
Definition at line 1607 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.784 `std::is_bind_expression< const volatile _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const volatile _Bind< _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.784.1 Detailed Description

```
template<typename _Signature>struct std::is_bind_expression< const volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

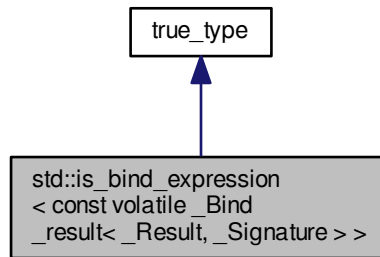
Definition at line 1591 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.785 `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.785.1 Detailed Description

```
template<typename _Result, typename _Signature>struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

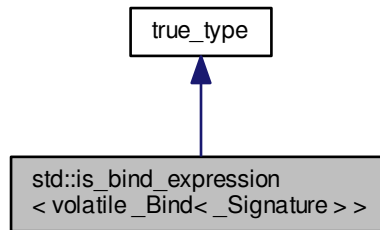
Definition at line 1623 of file `functional`.

The documentation for this struct was generated from the following file:

- `functional`

4.786 `std::is_bind_expression< volatile _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< volatile _Bind< _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.786.1 Detailed Description

```
template<typename _Signature>struct std::is_bind_expression< volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

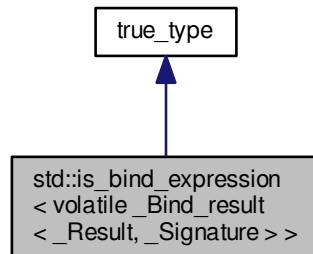
Definition at line 1583 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.787 `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.787.1 Detailed Description

```
template<typename _Result, typename _Signature>struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

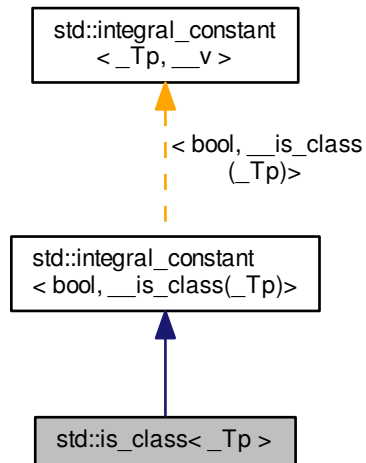
Definition at line 1615 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.788 std::is\_class&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_class< \_Tp >:



## Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.788.1 Detailed Description

```
template<typename _Tp>struct std::is_class< _Tp >
```

is\_class

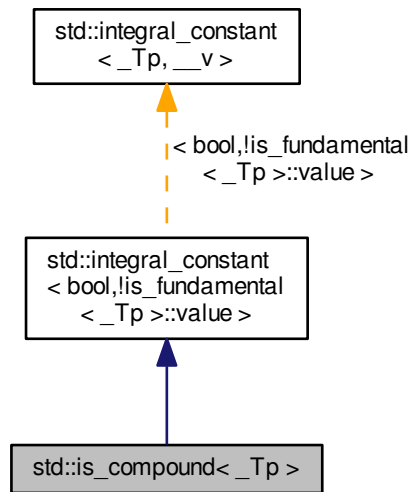
Definition at line 373 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.789 std::is\_compound&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_compound< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v > type`
- typedef `bool value_type`

## Public Member Functions

- constexpr `operator value_type ()`

## Static Public Attributes

- static constexpr `bool value`

## 4.789.1 Detailed Description

```
template<typename _Tp>struct std::is_compound< _Tp >
```

`is_compound`

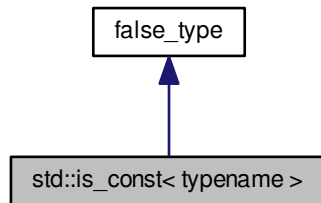
Definition at line 469 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

## 4.790 `std::is_const< typename >` Struct Template Reference

Inheritance diagram for `std::is_const< typename >`:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.790.1 Detailed Description

`template<typename> struct std::is_const< typename >`

`is_const`

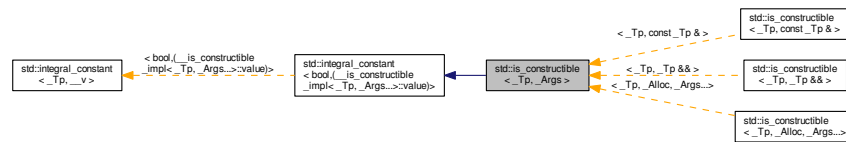
Definition at line 491 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.791 std::is\_constructible&lt; \_Tp, \_Args &gt; Struct Template Reference

Inheritance diagram for std::is\_constructible< \_Tp, \_Args >:



## Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.791.1 Detailed Description

template<typename \_Tp, typename... \_Args>struct std::is\_constructible< \_Tp, \_Args >

is\_constructible

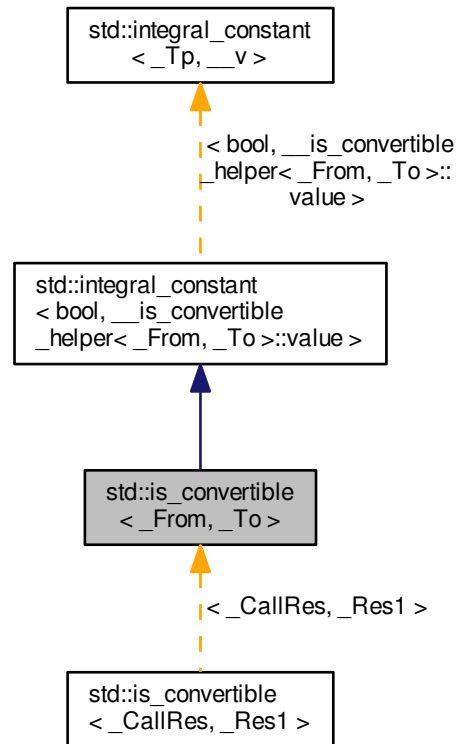
Definition at line 955 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.792 std::is\_convertible&lt; \_From, \_To &gt; Struct Template Reference

Inheritance diagram for std::is\_convertible< \_From, \_To >:



## Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.792.1 Detailed Description

```
template<typename _From, typename _To>struct std::is_convertible< _From, _To >
```

`is_convertible`

Definition at line 1317 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.793 `std::is_copy_assignable< _Tp >` Struct Template Reference

Inherits `__is_copy_assignable_impl< _Tp, bool >`.

## 4.793.1 Detailed Description

```
template<typename _Tp>struct std::is_copy_assignable< _Tp >
```

`is_copy_assignable`

Definition at line 1118 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.794 `std::is_copy_constructible< _Tp >` Struct Template Reference

Inherits `__is_copy_constructible_impl< _Tp, bool >`.

Inherited by `std::__is_copy_insertable< allocator< _Tp > >`.

## 4.794.1 Detailed Description

```
template<typename _Tp>struct std::is_copy_constructible< _Tp >
```

`is_copy_constructible`

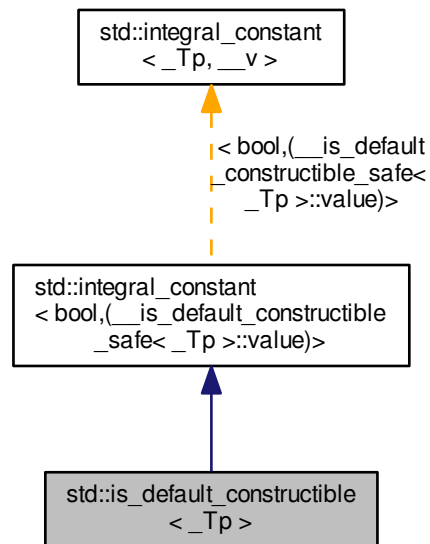
Definition at line 974 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.795 std::is\_default\_constructible&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_default\_constructible< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- `constexpr operator value_type ()`

## Static Public Attributes

- `static constexpr bool` **value**

## 4.795.1 Detailed Description

```
template<typename _Tp>struct std::is_default_constructible< _Tp >
```

`is_default_constructible`

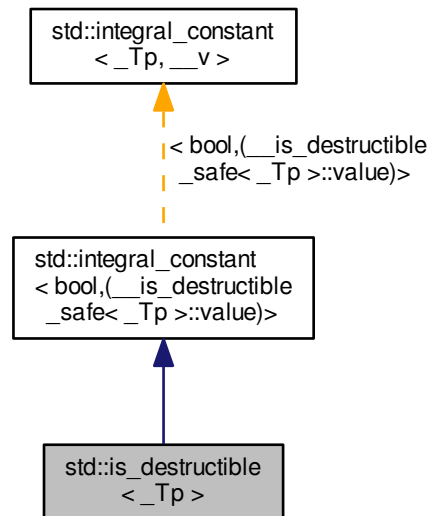
Definition at line 748 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.796 std::is\_destructible< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_destructible< \_Tp >:



### Public Types

- typedef [integral\\_constant](#) `< bool, __v >` **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

### 4.796.1 Detailed Description

```
template<typename _Tp> struct std::is_destructible< _Tp >
```

is\_destructible

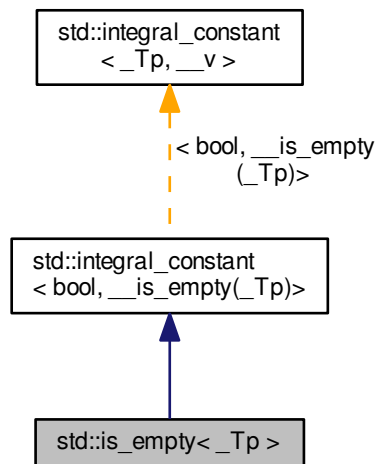
Definition at line 652 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.797 std::is\_empty< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_empty< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)  
    < bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 4.797.1 Detailed Description

```
template<typename _Tp>struct std::is_empty< _Tp >
```

is\_empty

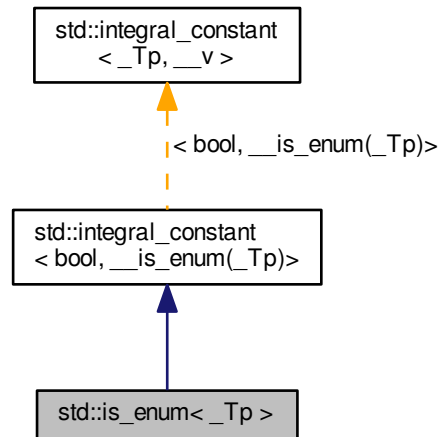
Definition at line 536 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.798 std::is\_enum< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_enum< \_Tp >:



##### Public Types

- typedef [integral\\_constant](#)  
    < bool, \_\_v > **type**
- typedef bool **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** ()

##### Static Public Attributes

- static constexpr bool **value**

## 4.798.1 Detailed Description

```
template<typename _Tp>struct std::is_enum< _Tp >
```

is\_enum

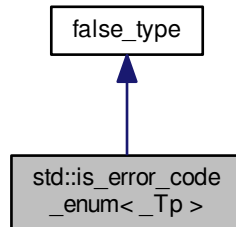
Definition at line 361 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.799 std::is\_error\_code\_enum&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_error\_code\_enum< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.799.1 Detailed Description

```
template<typename _Tp>struct std::is_error_code_enum< _Tp >
```

is\_error\_code\_enum

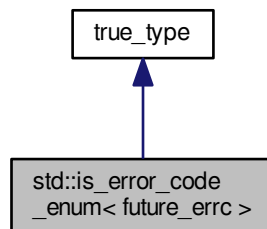
Definition at line 54 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 4.800 `std::is_error_code_enum< future_errc >` Struct Template Reference

Inheritance diagram for `std::is_error_code_enum< future_errc >`:



##### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** ()

##### Static Public Attributes

- static constexpr `_Tp` **value**

##### 4.800.1 Detailed Description

`template<> struct std::is_error_code_enum< future_errc >`

Specialization.

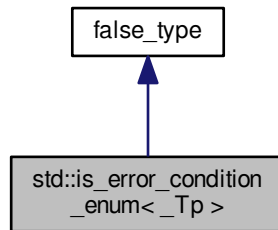
Definition at line 73 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

4.801 `std::is_error_condition_enum< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_error_condition_enum< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.801.1 Detailed Description

```
template<typename _Tp>struct std::is_error_condition_enum< _Tp >
```

`is_error_condition_enum`

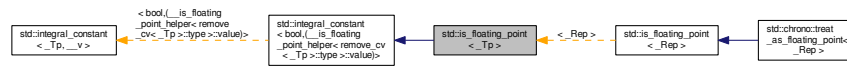
Definition at line 58 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 4.802 std::is\_floating\_point&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_floating\_point< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)  
    < bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.802.1 Detailed Description

```
template<typename _Tp>struct std::is_floating_point< _Tp >
```

`is_floating_point`

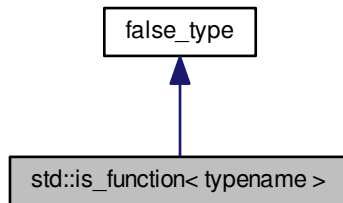
Definition at line 275 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.803 `std::is_function< typename >` Struct Template Reference

Inheritance diagram for `std::is_function< typename >`:



##### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** ()

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.803.1 Detailed Description

`template<typename> struct std::is_function< typename >`

`is_function`

Definition at line 379 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.804 `std::is_fundamental< _Tp >` Struct Template Reference

Inherits type< `is_arithmetic< _Tp >`, `is_void< _Tp >` >.

## 4.804.1 Detailed Description

```
template<typename _Tp>struct std::is_fundamental< _Tp >
```

is\_fundamental

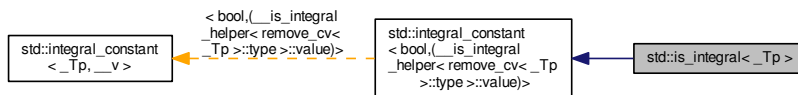
Definition at line 446 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.805 std::is\_integral&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_integral< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)  
`< bool, __v > type`
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.805.1 Detailed Description

```
template<typename _Tp>struct std::is_integral< _Tp >
```

is\_integral

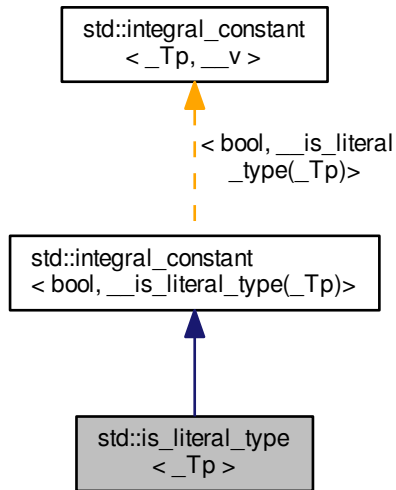
Definition at line 246 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.806 std::is\_literal\_type&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_literal\_type< \_Tp >:



## Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.806.1 Detailed Description

```
template<typename _Tp>struct std::is_literal_type< _Tp >
```

is\_literal\_type

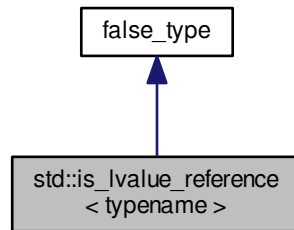
Definition at line 530 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.807 `std::is_lvalue_reference< typename >` Struct Template Reference

Inheritance diagram for `std::is_lvalue_reference< typename >`:

**Public Types**

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr `_Tp` **value**

## 4.807.1 Detailed Description

```
template<typename> struct std::is_lvalue_reference< typename >
```

`is_lvalue_reference`

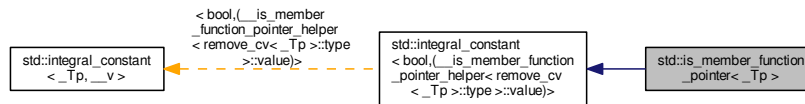
Definition at line 310 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.808 std::is\_member\_function\_pointer&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_member\_function\_pointer< \_Tp >:



## Public Types

- typedef [integral\\_constant](#) < bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.808.1 Detailed Description

```
template<typename _Tp>struct std::is_member_function_pointer< _Tp >
```

is\_member\_function\_pointer

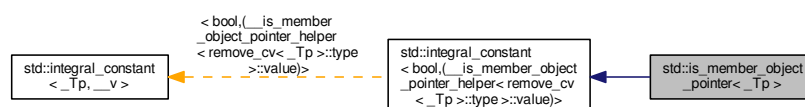
Definition at line 354 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.809 std::is\_member\_object\_pointer&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_member\_object\_pointer< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)  
< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.809.1 Detailed Description

template<typename \_Tp>struct std::is\_member\_object\_pointer< \_Tp >

is\_member\_object\_pointer

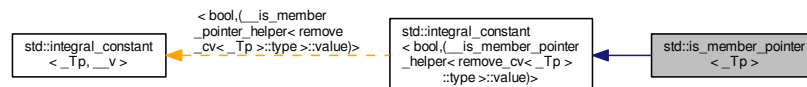
Definition at line 339 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.810 std::is\_member\_pointer&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_member\_pointer< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)  
< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.810.1 Detailed Description

```
template<typename _Tp>struct std::is_member_pointer< _Tp >
```

`is_member_pointer`

Definition at line 482 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.811 `std::is_move_assignable< _Tp >` Struct Template Reference

Inherits `__is_move_assignable_impl< _Tp, bool >`.

## 4.811.1 Detailed Description

```
template<typename _Tp>struct std::is_move_assignable< _Tp >
```

`is_move_assignable`

Definition at line 1136 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.812 `std::is_move_constructible< _Tp >` Struct Template Reference

Inherits `__is_move_constructible_impl< _Tp, bool >`.

## 4.812.1 Detailed Description

```
template<typename _Tp>struct std::is_move_constructible< _Tp >
```

`is_move_constructible`

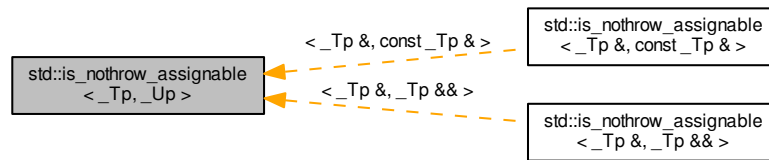
Definition at line 992 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.813 `std::is_nothrow_assignable<_Tp, _Up>` Struct Template Reference

Inheritance diagram for `std::is_nothrow_assignable<_Tp, _Up>`:



## 4.813.1 Detailed Description

```
template<typename _Tp, typename _Up>struct std::is_nothrow_assignable<_Tp, _Up>
```

`is_nothrow_assignable`

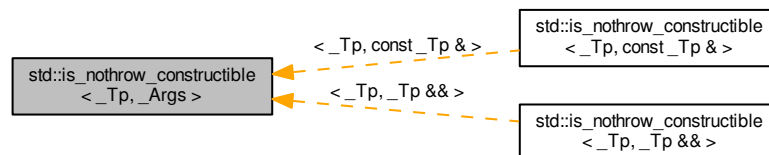
Definition at line 1147 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.814 `std::is_nothrow_constructible<_Tp, _Args>` Struct Template Reference

Inheritance diagram for `std::is_nothrow_constructible<_Tp, _Args>`:



## 4.814.1 Detailed Description

```
template<typename _Tp, typename... _Args>struct std::is_nothrow_constructible<_Tp, _Args>
```

`is_nothrow_constructible`

Definition at line 1041 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.815 `std::is_nothrow_copy_assignable< _Tp >` Struct Template Reference

Inherits `__is_nt_copy_assignable_impl< _Tp, bool >`.

##### 4.815.1 Detailed Description

```
template<typename _Tp>struct std::is_nothrow_copy_assignable< _Tp >
```

`is_nothrow_copy_assignable`

Definition at line 1166 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.816 `std::is_nothrow_copy_constructible< _Tp >` Struct Template Reference

Inherits `__is_nothrow_copy_constructible_impl< _Tp, bool >`.

##### 4.816.1 Detailed Description

```
template<typename _Tp>struct std::is_nothrow_copy_constructible< _Tp >
```

`is_nothrow_copy_constructible`

Definition at line 1060 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.817 `std::is_nothrow_default_constructible< _Tp >` Struct Template Reference

Inherits `type< is_default_constructible< _Tp >, __is_nt_default_constructible_impl< _Tp > >`.

Inherited by `std::__is_nt_constructible_impl< _Tp >`.

##### 4.817.1 Detailed Description

```
template<typename _Tp>struct std::is_nothrow_default_constructible< _Tp >
```

`is_nothrow_default_constructible`

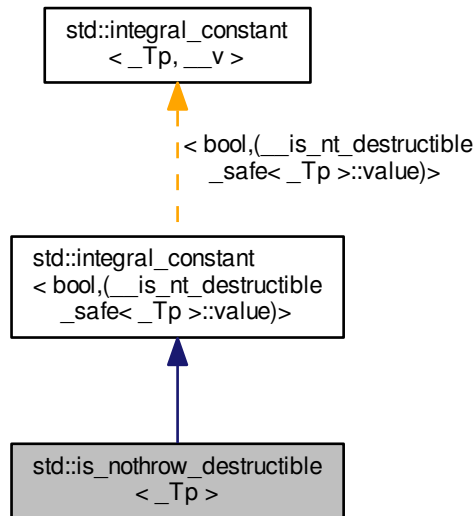
Definition at line 1018 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.818 std::is\_nothrow\_destructible&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_nothrow\_destructible< \_Tp >:



## Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `bool` **value**

## 4.818.1 Detailed Description

```
template<typename _Tp>struct std::is_nothrow_destructible< _Tp >
```

is\_nothrow\_destructible

Definition at line 700 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.819 `std::is_nothrow_move_assignable<_Tp>` Struct Template Reference

Inherits `__is_nt_move_assignable_impl<_Tp, bool>`.

##### 4.819.1 Detailed Description

```
template<typename _Tp>struct std::is_nothrow_move_assignable<_Tp>
```

`is_nothrow_move_assignable`

Definition at line 1184 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.820 `std::is_nothrow_move_constructible<_Tp>` Struct Template Reference

Inherits `__is_nothrow_move_constructible_impl<_Tp, bool>`.

##### 4.820.1 Detailed Description

```
template<typename _Tp>struct std::is_nothrow_move_constructible<_Tp>
```

`is_nothrow_move_constructible`

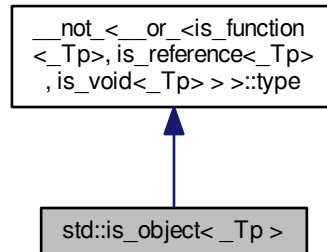
Definition at line 1078 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.821 std::is\_object&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_object< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.821.1 Detailed Description

```
template<typename _Tp>struct std::is_object< _Tp >
```

is\_object

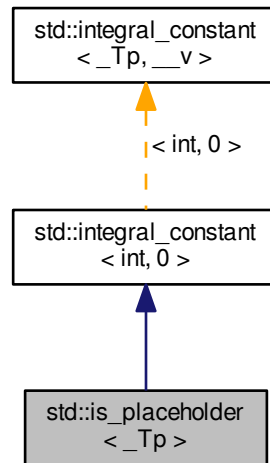
Definition at line 452 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.822 std::is\_placeholder&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_placeholder< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< int, \_\_v > **type**
- typedef int **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr int **value**

## 4.822.1 Detailed Description

```
template<typename _Tp>struct std::is_placeholder< _Tp >
```

Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].

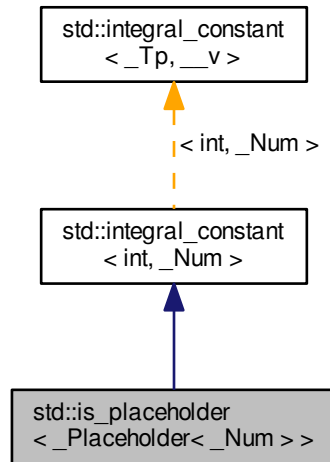
Definition at line 982 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.823 `std::is_placeholder< _Placeholder< _Num > >` Struct Template Reference

Inheritance diagram for `std::is_placeholder< _Placeholder< _Num > >`:



## Public Types

- typedef `integral_constant< int, __v > type`
- typedef int `value_type`

## Public Member Functions

- constexpr `operator value_type ()`

## Static Public Attributes

- static constexpr int `value`

## 4.823.1 Detailed Description

`template<int _Num>struct std::is_placeholder< _Placeholder< _Num > >`

Partial specialization of `is_placeholder` that provides the placeholder number for the placeholder objects defined by `libstdc++`.

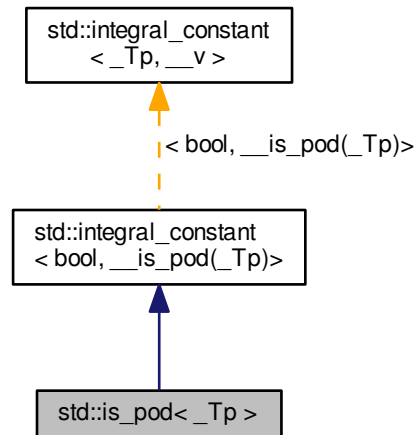
Definition at line 1044 of file `functional`.

The documentation for this struct was generated from the following file:

- `functional`

## 4.824 std::is\_pod&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_pod< \_Tp >:



## Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.824.1 Detailed Description

```
template<typename _Tp>struct std::is_pod< _Tp >
```

is\_pod

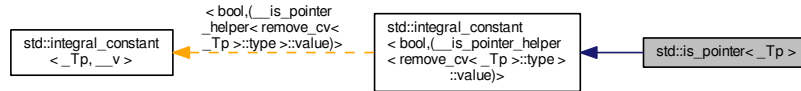
Definition at line 524 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.825 std::is\_pointer&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_pointer< \_Tp >:



## Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool `value_type`

## Public Member Functions

- constexpr `operator value_type ()`

## Static Public Attributes

- static constexpr bool `value`

## 4.825.1 Detailed Description

```
template<typename _Tp>struct std::is_pointer< _Tp >
```

is\_pointer

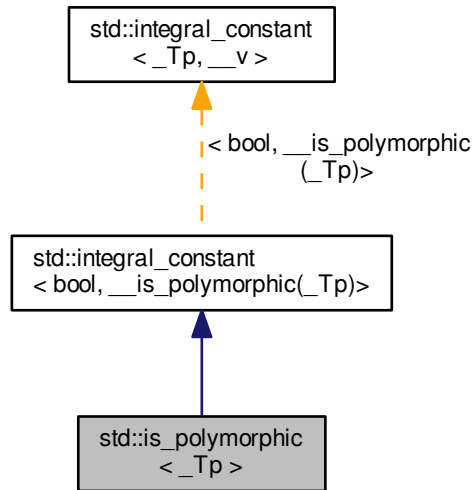
Definition at line 303 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.826 std::is\_polymorphic&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_polymorphic< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v > type`
- typedef `bool value_type`

## Public Member Functions

- `constexpr operator value_type ()`

## Static Public Attributes

- `static constexpr bool value`

## 4.826.1 Detailed Description

```
template<typename _Tp>struct std::is_polymorphic< _Tp >
```

`is_polymorphic`

Definition at line 542 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

## 4.827 `std::is_reference< _Tp >` Struct Template Reference

Inherits type< `is_lvalue_reference< _Tp >`, `is_rvalue_reference< _Tp >` >.

### 4.827.1 Detailed Description

```
template<typename _Tp>struct std::is_reference< _Tp >
```

`is_reference`

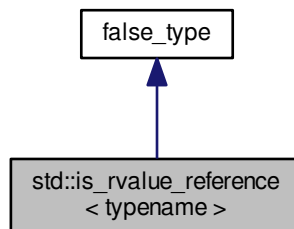
Definition at line 433 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.828 `std::is_rvalue_reference< typename >` Struct Template Reference

Inheritance diagram for `std::is_rvalue_reference< typename >`:



### Public Types

- typedef [integral\\_constant< \\_Tp, \\_\\_v >](#) **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

## 4.828.1 Detailed Description

```
template<typename>struct std::is_rvalue_reference< typename >
```

`is_rvalue_reference`

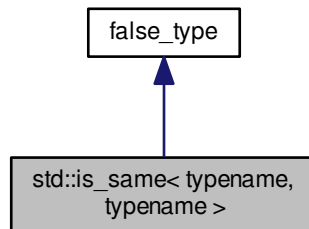
Definition at line 319 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.829 `std::is_same< typename, typename >` Struct Template Reference

Inheritance diagram for `std::is_same< typename, typename >`:



## Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.829.1 Detailed Description

```
template<typename, typename>struct std::is_same< typename, typename >
```

`is_same`

Definition at line 1278 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.830 std::is\_scalar< \_Tp > Struct Template Reference

Inherits type< is\_arithmetic< \_Tp >, is\_enum< \_Tp >, is\_pointer< \_Tp >, is\_member\_pointer< \_Tp >, \_\_is\_nullptr\_t< \_Tp > >.

##### 4.830.1 Detailed Description

```
template<typename _Tp>struct std::is_scalar< _Tp >
```

is\_scalar

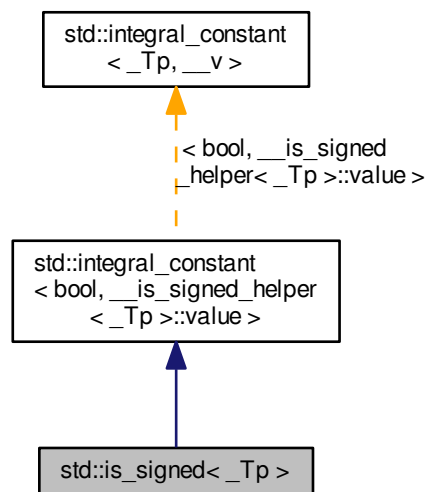
Definition at line 462 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.831 std::is\_signed< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_signed< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)  
    < bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** ()

#### Static Public Attributes

- static constexpr bool **value**

#### 4.831.1 Detailed Description

`template<typename _Tp>struct std::is_signed<_Tp>`

`is_signed`

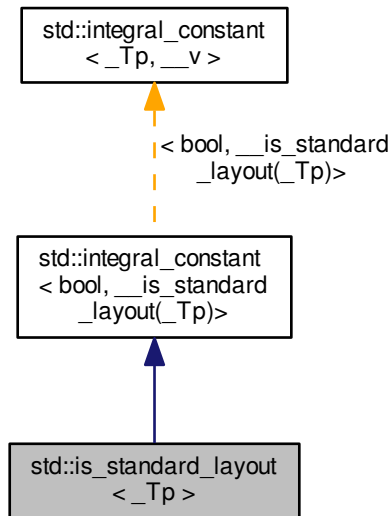
Definition at line 569 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.832 std::is\_standard\_layout&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_standard\_layout< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `bool` **value**

## 4.832.1 Detailed Description

```
template<typename _Tp>struct std::is_standard_layout< _Tp >
```

`is_standard_layout`

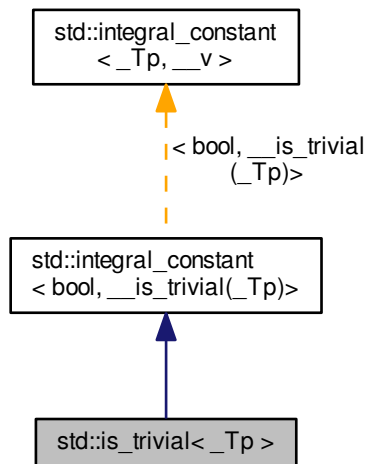
Definition at line 517 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.833 std::is\_trivial< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_trivial< \_Tp >:



##### Public Types

- typedef [integral\\_constant](#)  
    < bool, \_\_v > **type**
- typedef bool **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** ()

##### Static Public Attributes

- static constexpr bool **value**

##### 4.833.1 Detailed Description

```
template<typename _Tp>struct std::is_trivial< _Tp >
```

is\_trivial

Definition at line 509 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.834 std::is\_trivially\_destructible< \_Tp > Struct Template Reference

Inherits type< is\_destructible< \_Tp >, integral\_constant< bool, \_\_has\_trivial\_destructor(\_Tp)> >.

##### 4.834.1 Detailed Description

```
template<typename _Tp>struct std::is_trivially_destructible< _Tp >
```

is\_trivially\_constructible (still unimplemented)

is\_trivially\_default\_constructible (still unimplemented) is\_trivially\_copy\_constructible (still unimplemented) is\_trivially\_move\_constructible (still unimplemented) is\_trivially\_assignable (still unimplemented) is\_trivially\_copy\_assignable (still unimplemented) is\_trivially\_move\_assignable (still unimplemented) is\_trivially\_destructible

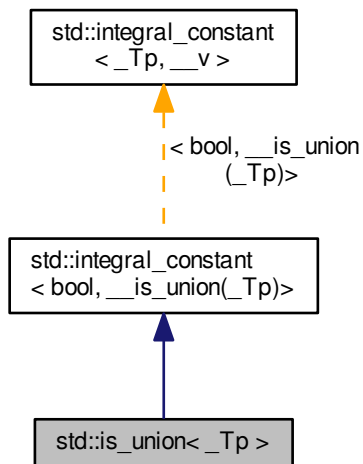
Definition at line 1204 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.835 std::is\_union< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_union< \_Tp >:



##### Public Types

- typedef [integral\\_constant](#)  
`< bool, __v > type`

- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** ()

#### Static Public Attributes

- static constexpr bool **value**

#### 4.835.1 Detailed Description

template<typename \_Tp>struct std::is\_union< \_Tp >

is\_union

Definition at line 367 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.836 std::is\_unsigned< \_Tp > Struct Template Reference

Inherits type< is\_arithmetic< \_Tp >, \_\_not\_< is\_signed< \_Tp > > >.

#### 4.836.1 Detailed Description

template<typename \_Tp>struct std::is\_unsigned< \_Tp >

is\_unsigned

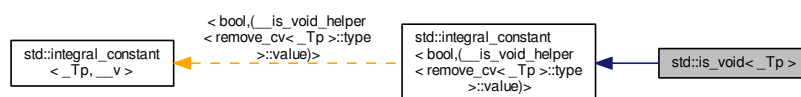
Definition at line 575 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.837 std::is\_void< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_void< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)  
    < bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 4.837.1 Detailed Description

template<typename \_Tp>struct std::is\_void< \_Tp >

is\_void

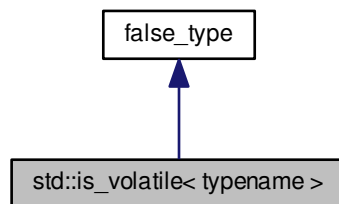
Definition at line 163 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.838 `std::is_volatile< typename >` Struct Template Reference

Inheritance diagram for `std::is_volatile< typename >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp,  
    \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.838.1 Detailed Description

template<typename>struct std::is\_volatile< typename >

is\_volatile

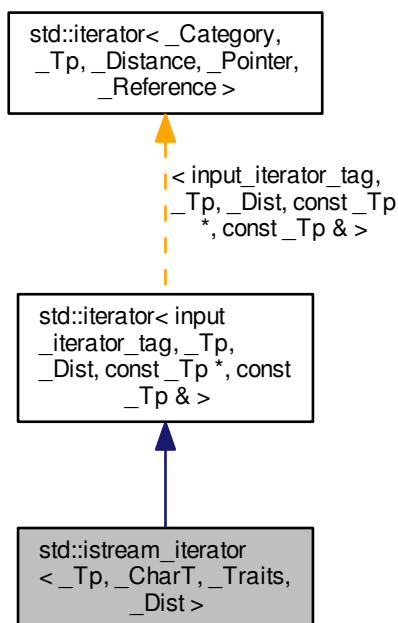
Definition at line 500 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.839 std::istream\_iterator&lt; \_Tp, \_CharT, \_Traits, \_Dist &gt; Class Template Reference

Inheritance diagram for std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Dist` **difference\_type**
- typedef `basic_istream< _CharT, _Traits >` **istream\_type**
- typedef `input_iterator_tag` **iterator\_category**
- typedef `const _Tp *` **pointer**
- typedef `const _Tp &` **reference**
- typedef `_Traits` **traits\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr `istream_iterator` ()
- `istream_iterator` (`istream_type` &\_\_s)
- `istream_iterator` (const `istream_iterator` &\_\_obj)
- `bool` **\_M\_equal** (const `istream_iterator` &\_\_x) const
- `const _Tp &` **operator\*** () const
- `istream_iterator` & **operator++** ()
- `istream_iterator` **operator++** (int)
- `const _Tp *` **operator->** () const

## 4.839.1 Detailed Description

template<typename `_Tp`, typename `_CharT` = `char`, typename `_Traits` = `char_traits<_CharT>`, typename `_Dist` = `ptrdiff_t`>class std::istream\_iterator< `_Tp`, `_CharT`, `_Traits`, `_Dist` >

Provides input iterator semantics for streams.

Definition at line 49 of file `stream_iterator.h`.

## 4.839.2 Member Typedef Documentation

4.839.2.1 typedef `_Dist` std::iterator< `input_iterator_tag` , `_Tp`, `_Dist` , `const _Tp *` , `const _Tp &` >::**difference\_type**  
[*inherited*]

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.839.2.2 typedef `input_iterator_tag` std::iterator< `input_iterator_tag` , `_Tp`, `_Dist` , `const _Tp *` , `const _Tp &` >::**iterator\_category** [*inherited*]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.839.2.3 typedef `const _Tp *` std::iterator< `input_iterator_tag` , `_Tp`, `_Dist` , `const _Tp *` , `const _Tp &` >::**pointer**  
[*inherited*]

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.839.2.4 `typedef const _Tp & std::iterator< input_iterator_tag, _Tp, _Dist, const _Tp *, const _Tp & >::reference`  
`[inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

4.839.2.5 `typedef _Tp std::iterator< input_iterator_tag, _Tp, _Dist, const _Tp *, const _Tp & >::value_type`  
`[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

#### 4.839.3 Constructor & Destructor Documentation

4.839.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>`  
`constexpr std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( ) [inline]`

Construct end of input stream iterator.

Definition at line 64 of file stream\_iterator.h.

4.839.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>`  
`std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( istream_type & __s ) [inline]`

Construct start of input stream iterator.

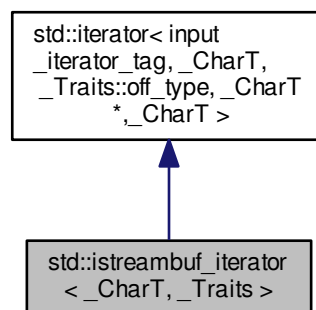
Definition at line 68 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

#### 4.840 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::istreambuf\_iterator< \_CharT, \_Traits >:



## Public Types

- typedef `_Distance` [difference\\_type](#)
- typedef `_Category` [iterator\\_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value\\_type](#)
  
- typedef `_CharT` [char\\_type](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `_Traits::int_type` [int\\_type](#)
- typedef [basic\\_streambuf](#)  
`<_CharT, _Traits>` [streambuf\\_type](#)
- typedef [basic\\_istream](#)`<_CharT,`  
`_Traits>` [istream\\_type](#)

## Public Member Functions

- constexpr [istreambuf\\_iterator](#) () noexcept
- **[istreambuf\\_iterator](#)** (const [istreambuf\\_iterator](#) &) noexcept=default
- [istreambuf\\_iterator](#) ([istream\\_type](#) &\_\_s) noexcept
- [istreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- bool [equal](#) (const [istreambuf\\_iterator](#) &\_\_b) const
- [char\\_type](#) [operator\\*](#) () const
- [istreambuf\\_iterator](#) & [operator++](#) ()
- [istreambuf\\_iterator](#) [operator++](#) (int)

## Friends

- template<bool \_IsMove, typename \_CharT2 >  
`__gnu_cxx::__enable_if`  
`<__is_char<_CharT2>`  
`::_value, _CharT2 * >::_type` [\\_\\_copy\\_move\\_a2](#) ([istreambuf\\_iterator](#)`<_CharT2>`, [istreambuf\\_iterator](#)`<_`  
`CharT2>`, `_CharT2 *`)
- template<typename \_CharT2 >  
`__gnu_cxx::__enable_if`  
`<__is_char<_CharT2>`  
`::_value, ostreambuf_iterator`  
`<_CharT2> >::_type` [copy](#) ([istreambuf\\_iterator](#)`<_CharT2>`, [istreambuf\\_iterator](#)`<_CharT2>`, [ostreambuf-](#)  
`iterator<_CharT2>)`
- template<typename \_CharT2 >  
`__gnu_cxx::__enable_if`  
`<__is_char<_CharT2>`  
`::_value, istreambuf_iterator`  
`<_CharT2> >::_type` [find](#) ([istreambuf\\_iterator](#)`<_CharT2>`, [istreambuf\\_iterator](#)`<_CharT2>`, const `_CharT2`  
&)

## 4.840.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::istreambuf_iterator< _CharT, _Traits >
```

Provides input iterator semantics for streambufs.

Definition at line 50 of file streambuf\_iterator.h.

## 4.840.2 Member Typedef Documentation

```
4.840.2.1 template<typename _CharT, typename _Traits > typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

Definition at line 64 of file streambuf\_iterator.h.

```
4.840.2.2 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type [inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

```
4.840.2.3 template<typename _CharT, typename _Traits > typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
```

Public typedefs.

Definition at line 66 of file streambuf\_iterator.h.

```
4.840.2.4 template<typename _CharT, typename _Traits > typedef basic_istream<_CharT, _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istream_type
```

Public typedefs.

Definition at line 68 of file streambuf\_iterator.h.

```
4.840.2.5 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

```
4.840.2.6 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename
_Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

4.840.2.7 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference`  
[inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.840.2.8 `template<typename _CharT, typename _Traits> typedef basic_streambuf<_CharT, _Traits>  
std::istreambuf_iterator< _CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 67 of file `streambuf_iterator.h`.

4.840.2.9 `template<typename _CharT, typename _Traits> typedef _Traits std::istreambuf_iterator< _CharT, _Traits  
>::traits_type`

Public typedefs.

Definition at line 65 of file `streambuf_iterator.h`.

4.840.2.10 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type`  
[inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

#### 4.840.3 Constructor & Destructor Documentation

4.840.3.1 `template<typename _CharT, typename _Traits> constexpr std::istreambuf_iterator< _CharT, _Traits  
>::istreambuf_iterator( ) [inline], [noexcept]`

Construct end of input stream iterator.

Definition at line 102 of file `streambuf_iterator.h`.

4.840.3.2 `template<typename _CharT, typename _Traits> std::istreambuf_iterator< _CharT, _Traits  
>::istreambuf_iterator( istream_type & __s ) [inline], [noexcept]`

Construct start of input stream iterator.

Definition at line 112 of file `streambuf_iterator.h`.

4.840.3.3 `template<typename _CharT, typename _Traits> std::istreambuf_iterator< _CharT, _Traits  
>::istreambuf_iterator( streambuf_type * __s ) [inline], [noexcept]`

Construct start of streambuf iterator.

Definition at line 116 of file `streambuf_iterator.h`.

#### 4.840.4 Member Function Documentation

4.840.4.1 `template<typename _CharT, typename _Traits> bool std::istreambuf_iterator< _CharT, _Traits >::equal( const istreambuf_iterator< _CharT, _Traits > & __b ) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 172 of file streambuf\_iterator.h.

4.840.4.2 `template<typename _CharT, typename _Traits> char_type std::istreambuf_iterator< _CharT, _Traits >::operator*( ) const [inline]`

Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator\*() on an end of stream is undefined.

Definition at line 123 of file streambuf\_iterator.h.

4.840.4.3 `template<typename _CharT, typename _Traits> istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits >::operator++( ) [inline]`

Advance the iterator. Calls streambuf.sbumpc().

Definition at line 137 of file streambuf\_iterator.h.

References std::basic\_streambuf< \_CharT, \_Traits >::sbumpc().

4.840.4.4 `template<typename _CharT, typename _Traits> istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++( int ) [inline]`

Advance the iterator. Calls streambuf.sbumpc().

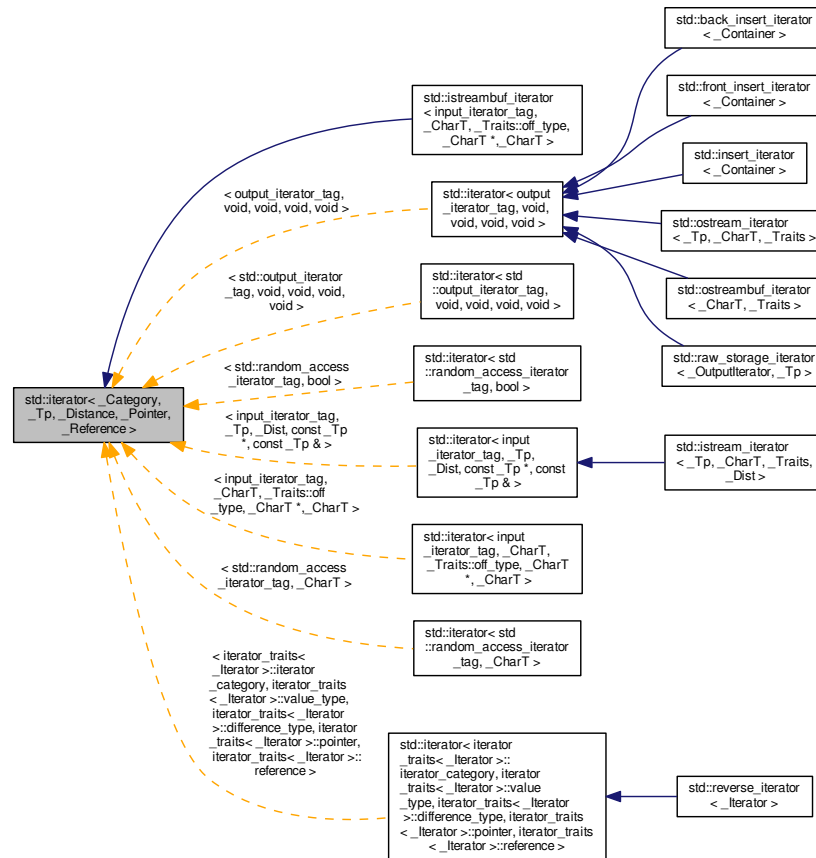
Definition at line 152 of file streambuf\_iterator.h.

References std::basic\_streambuf< \_CharT, \_Traits >::sbumpc().

The documentation for this class was generated from the following file:

- [streambuf\\_iterator.h](#)

#### 4.841 `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>` Struct Template Reference



## Public Types

#### 4.841.1 Detailed Description

Common iterator class.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 118 of file `stl_iterator_base_types.h`.

#### 4.841.2 Member Typedef Documentation

4.841.2.1 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,  
typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference  
>::difference_type`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.841.2.2 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,  
typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference  
>::iterator_category`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.841.2.3 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename  
_Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.841.2.4 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename  
_Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference`

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.841.2.5 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename  
_Reference = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.842 `std::iterator_traits<_Tp * >` Struct Template Reference

### Public Types

- typedef `ptrdiff_t` **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `_Tp` **value\_type**

## 4.842.1 Detailed Description

```
template<typename _Tp>struct std::iterator_traits< _Tp * >
```

Partial specialization for pointer types.

Definition at line 175 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

4.843 `std::iterator_traits< const _Tp * >` Struct Template Reference

## Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

## 4.843.1 Detailed Description

```
template<typename _Tp>struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

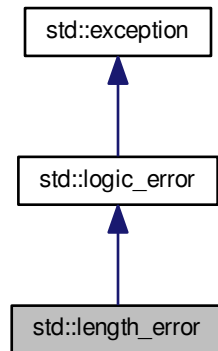
Definition at line 186 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.844 `std::length_error` Class Reference

Inheritance diagram for `std::length_error`:



### Public Member Functions

- **`length_error`** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

#### 4.844.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a `basic_string` instance).

Definition at line 91 of file `stdexcept`.

#### 4.844.2 Member Function Documentation

##### 4.844.2.1 virtual const char\* `std::logic_error::what` ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

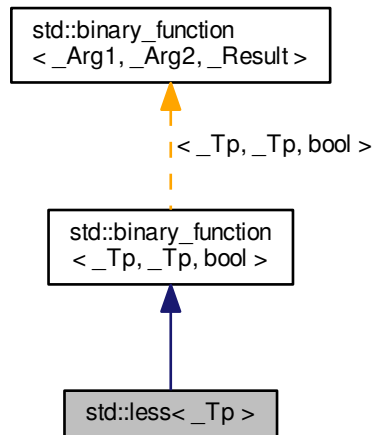
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 4.845 std::less&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::less< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.845.1 Detailed Description

```
template<typename _Tp>struct std::less< _Tp >
```

One of the [comparison functors](#).

Definition at line 231 of file `stl_function.h`.

## 4.845.2 Member Typedef Documentation

4.845.2.1 typedef `_Tp` `std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.845.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.845.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

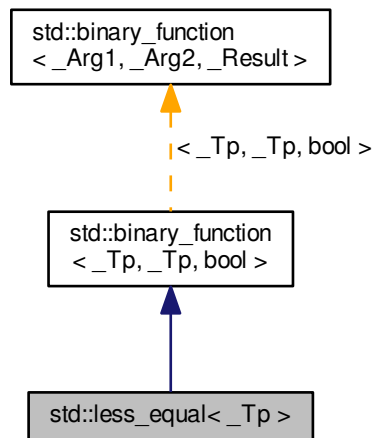
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.846 std::less\_equal< \_Tp > Struct Template Reference

Inheritance diagram for `std::less_equal< _Tp >`:



### Public Types

- `typedef _Tp` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (`const _Tp &__x, const _Tp &__y`) `const`

## 4.846.1 Detailed Description

```
template<typename _Tp>struct std::less_equal<_Tp>
```

One of the [comparison functors](#).

Definition at line 249 of file `stl_function.h`.

## 4.846.2 Member Typedef Documentation

4.846.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.846.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.846.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.847 `std::linear_congruential_engine<_UIntType, __a, __c, __m>` Class Template Reference

## Public Types

- `typedef _UIntType result_type`

## Public Member Functions

- [linear\\_congruential\\_engine](#) ([result\\_type](#) \_\_s=default\_seed)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type>`  
[linear\\_congruential\\_engine](#) (\_Sseq &\_\_q)
- `void discard` (unsigned long long \_\_z)
- [result\\_type](#) `operator()` ()
- `void seed` ([result\\_type](#) \_\_s=default\_seed)
- `template<typename _Sseq>`  
`std::enable_if< std::is_class`  
`<_Sseq>::value>::type` [seed](#) (\_Sseq &\_\_q)

## Static Public Member Functions

- `static constexpr result_type` [max](#) ()
- `static constexpr result_type` [min](#) ()

## Static Public Attributes

- static constexpr [result\\_type](#) **default\_seed**
- static constexpr [result\\_type](#) **increment**
- static constexpr [result\\_type](#) **modulus**
- static constexpr [result\\_type](#) **multiplier**

## Friends

- template<typename \_UIntType1 , \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT , typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT,  
\_Traits > & [operator<<](#) ( [std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::linear\\_congruential\\_engine](#)<  
\_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &\_\_lcr)
- bool [operator==](#) (const [linear\\_congruential\\_engine](#) &\_\_lhs, const [linear\\_congruential\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1 , \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT , typename \_Traits >  
[std::basic\\_istream](#)< \_CharT,  
\_Traits > & [operator>>](#) ( [std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::linear\\_congruential\\_engine](#)< \_UInt-  
Type1, \_\_a1, \_\_c1, \_\_m1 > &\_\_lcr)

## 4.847.1 Detailed Description

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>class std::linear_congruential_engine< _UIntType, __a,
__c, __m >
```

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 241 of file random.h.

## 4.847.2 Member Typedef Documentation

```
4.847.2.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> typedef _UIntType
std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

Definition at line 244 of file random.h.

## 4.847.3 Constructor &amp; Destructor Documentation

```
4.847.3.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> std::linear_congruential_engine<
_UIntType, __a, __c, __m >::linear_congruential_engine (result_type __s = default_seed) [inline],
[explicit]
```

Constructs a `linear_congruential_engine` random number generator engine with seed `__s`. The default seed value is 1.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The initial seed value. |
|------------------|-------------------------|

Definition at line 268 of file random.h.

References `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

```
4.847.3.2 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq,
typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value> ::type>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (_Sseq & __q)
[inline], [explicit]
```

Constructs a `linear_congruential_engine` random number generator engine seeded from the seed sequence `__q`.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Definition at line 281 of file random.h.

References `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

## 4.847.4 Member Function Documentation

```
4.847.4.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear-
_congruential_engine< _UIntType, __a, __c, __m >::discard (unsigned long long __z)
[inline]
```

Discard a sequence of random numbers.

Definition at line 325 of file random.h.

```
4.847.4.2 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type
std::linear_congruential_engine< _UIntType, __a, __c, __m >::max () [inline], [static]
```

Gets the largest possible value in the output range.

Definition at line 318 of file random.h.

```
4.847.4.3 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type
std::linear_congruential_engine< _UIntType, __a, __c, __m >::min () [inline], [static]
```

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be  $> 0$ , otherwise 0 is allowed.

Definition at line 311 of file random.h.

```
4.847.4.4 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type
std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator()() [inline]
```

Gets the next random number in the sequence.

Definition at line 335 of file random.h.

4.847.4.5 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed ( result_type __s = default_seed )`

Reseeds the linear\_congruential\_engine random number generator engine sequence to the seed \_\_s.

#### Parameters

|                  |               |
|------------------|---------------|
| <code>__s</code> | The new seed. |
|------------------|---------------|

Seeds the LCR with integral value \_\_s, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 120 of file bits/random.tcc.

Referenced by std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >::linear\_congruential\_engine().

4.847.4.6 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq > std::enable_if< std::is_class< _Sseq >::value >::type std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed ( _Sseq & __q )`

Reseeds the linear\_congruential\_engine random number generator engine sequence using values from the seed sequence \_\_q.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Seeds the LCR engine with a value generated by \_\_q.

Definition at line 136 of file bits/random.tcc.

References std::\_\_lg().

### 4.847.5 Friends And Related Function Documentation

4.847.5.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits> & operator<< ( std::basic_ostream< _CharT, _Traits> & __os, const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1> & __lcr ) [friend]`

Writes the textual representation of the state x(i) of x to \_\_os.

#### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__os</code>  | The output stream.                                      |
| <code>__lcr</code> | A % linear_congruential_engine random number generator. |

#### Returns

`__os`.

4.847.5.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool operator== ( const linear_congruential_engine< _UIntType, __a, __c, __m> & __lhs, const linear_congruential_engine< _UIntType, __a, __c, __m> & __rhs ) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

## Parameters

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__lhs</code> | A linear congruential random number generator object.       |
| <code>__rhs</code> | Another linear congruential random number generator object. |

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 353 of file random.h.

4.847.5.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1> & __lcr ) [friend]`

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__is</code>  | The input stream.                                       |
| <code>__lcr</code> | A % linear_congruential_engine random number generator. |

## Returns

`__is`.

## 4.847.6 Member Data Documentation

4.847.6.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m>::increment [static]`

An increment.

Definition at line 255 of file random.h.

4.847.6.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m>::modulus [static]`

The modulus.

Definition at line 257 of file random.h.

4.847.6.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m>::multiplier [static]`

The multiplier.

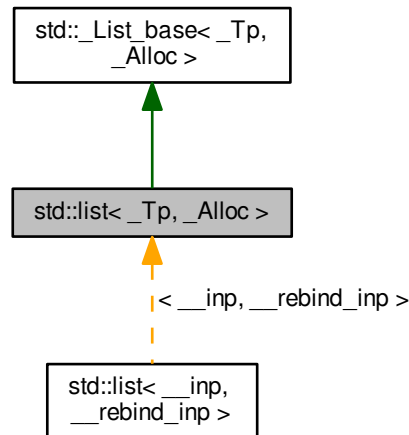
Definition at line 253 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.848 std::list&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::list< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_List_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `list()`
- `list(const allocator_type &__a)`
- `list(size_type __n)`
- `list(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`

- [list](#) (const [list](#) &\_\_x)
- [list](#) ([list](#) &&\_\_x) noexcept
- [list](#) ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
[list](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) ([initializer\\_list](#)< value\_type > \_\_l)
- reference [back](#) ()
- const\_reference [back](#) () const
- [iterator begin](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- [const\\_reverse\\_iterator crbegin](#) () const noexcept
- [const\\_reverse\\_iterator crend](#) () const noexcept
- template<typename... \_Args>  
[iterator emplace](#) ([iterator](#) \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_front](#) (\_Args &&...\_\_args)
- bool [empty](#) () const noexcept
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [iterator erase](#) ([iterator](#) \_\_position)
- [iterator erase](#) ([iterator](#) \_\_first, [iterator](#) \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- [iterator insert](#) ([iterator](#) \_\_position, const value\_type &\_\_x)
- [iterator insert](#) ([iterator](#) \_\_position, value\_type &&\_\_x)
- void [insert](#) ([iterator](#) \_\_p, [initializer\\_list](#)< value\_type > \_\_l)
- void [insert](#) ([iterator](#) \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void [insert](#) ([iterator](#) \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type [max\\_size](#) () const noexcept
- void [merge](#) ([list](#) &&\_\_x)
- void [merge](#) ([list](#) &\_\_x)
- template<typename \_StrictWeakOrdering >  
void [merge](#) ([list](#) &&\_\_x, \_StrictWeakOrdering \_\_comp)
- template<typename \_StrictWeakOrdering >  
void [merge](#) ([list](#) &\_\_x, \_StrictWeakOrdering \_\_comp)
- [list & operator=](#) (const [list](#) &\_\_x)
- [list & operator=](#) ([list](#) &&\_\_x)
- [list & operator=](#) ([initializer\\_list](#)< value\_type > \_\_l)
- void [pop\\_back](#) ()
- void [pop\\_front](#) ()
- void [push\\_back](#) (const value\_type &\_\_x)

- void **push\_back** (value\_type && \_\_x)
- void **push\_front** (const value\_type & \_\_x)
- void **push\_front** (value\_type && \_\_x)
- **reverse\_iterator** **rbegin** () noexcept
- **const\_reverse\_iterator** **rbegin** () const noexcept
- void **remove** (const \_Tp & \_\_value)
- template<typename \_Predicate >  
void **remove\_if** (\_Predicate)
- **reverse\_iterator** **rend** () noexcept
- **const\_reverse\_iterator** **rend** () const noexcept
- void **resize** (size\_type \_\_new\_size)
- void **resize** (size\_type \_\_new\_size, const value\_type & \_\_x)
- void **reverse** () noexcept
- size\_type **size** () const noexcept
- void **sort** ()
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering)
- void **splice** (iterator \_\_position, list && \_\_x)
- void **splice** (iterator \_\_position, list & \_\_x)
- void **splice** (iterator \_\_position, list && \_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list & \_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list && \_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_position, list & \_\_x, iterator \_\_first, iterator \_\_last)
- void **swap** (list & \_\_x)
- void **unique** ()
- template<typename \_BinaryPredicate >  
void **unique** (\_BinaryPredicate)

### Protected Types

- typedef **\_List\_node**< \_Tp > **\_Node**

### Protected Member Functions

- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_check\_equal\_allocators** (list & \_\_x)
- void **\_M\_clear** ()
- template<typename... \_Args>  
**\_Node** \* **\_M\_create\_node** (\_Args &&... \_\_args)
- void **\_M\_default\_append** (size\_type \_\_n)
- void **\_M\_default\_initialize** (size\_type \_\_n)
- void **\_M\_erase** (iterator \_\_position)
- void **\_M\_fill\_assign** (size\_type \_\_n, const value\_type & \_\_val)
- void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type & \_\_x)
- **\_List\_node**< \_Tp > \* **\_M\_get\_node** ()
- **\_Node\_alloc\_type** & **\_M\_get\_Node\_allocator** () noexcept
- const **\_Node\_alloc\_type** & **\_M\_get\_Node\_allocator** () const noexcept

- `_Tp_alloc_type _M_get_Tp_allocator ()` const noexcept
- `void _M_init ()`
- `template<typename _Integer>`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator>`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename... _Args>`  
`void _M_insert (iterator __position, _Args &&... __args)`
- `void _M_put_node (_List_node<_Tp> * __p)`
- `void _M_transfer (iterator __position, iterator __first, iterator __last)`

#### Protected Attributes

- `_List_impl _M_impl`

#### 4.848.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::list<_Tp, _Alloc>`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

A <----> B <----> C <----> D

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's next/previous pointers refer to itself, the list is empty.

Definition at line 438 of file `stl_list.h`.

#### 4.848.2 Constructor & Destructor Documentation

4.848.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( ) [inline]`

Default constructor creates no elements.

Definition at line 523 of file `stl_list.h`.

4.848.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( const allocator_type &__a ) [inline],[explicit]`

Creates a list with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

Definition at line 531 of file `stl_list.h`.

4.848.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( size_type __n ) [inline],[explicit]`

Creates a list with default constructed elements.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
|------------------|---------------------------------------------|

This constructor fills the list with `__n` default constructed elements.

Definition at line 543 of file `stl_list.h`.

4.848.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( size_type __n, const value_type &__value, const allocator_type &__a = allocator_type() ) [inline]`

Creates a list with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator object.                        |

This constructor fills the list with `__n` copies of `__value`.

Definition at line 555 of file `stl_list.h`.

4.848.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( const list<_Tp, _Alloc> &__x ) [inline]`

List copy constructor.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The newly-created list uses a copy of the allocation object used by `__x`.

Definition at line 582 of file `stl_list.h`.

4.848.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( list<_Tp, _Alloc> && __x ) [inline], [noexcept]`

List move constructor.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The newly-created list contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified list.

Definition at line 594 of file `stl_list.h`.

4.848.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( initializer_list<value_type> & __l, const allocator_type & __a = allocator_type() ) [inline]`

Builds a list from an `initializer_list`.

#### Parameters

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |
| <code>__a</code> | An allocator object.                                          |

Create a list consisting of copies of the elements in the `initializer_list` `__l`. This is linear in `__l.size()`.

Definition at line 605 of file `stl_list.h`.

4.848.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> std::list<_Tp, _Alloc>::list ( _InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type() ) [inline]`

Builds a list from a range.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__a</code>     | An allocator object. |

Create a list consisting of copies of the elements from `[__first, __last)`. This is linear in `N` (where `N` is `distance(__first, __last)`).

Definition at line 624 of file `stl_list.h`.

### 4.848.3 Member Function Documentation

4.848.3.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> _Node* std::list<_Tp, _Alloc>::M.create_node ( _Args &&... __args ) [inline], [protected]`

#### Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__args</code> | An instance of user data. |
|---------------------|---------------------------|

Allocates space for a new node and constructs a copy of `__args` in it.

Definition at line 500 of file `stl_list.h`.

4.848.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::assign ( size_type __n, const value_type & __val ) [inline]`

Assigns a given value to a list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a list with `__n` copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 702 of file `stl_list.h`.

Referenced by `std::list<__inp, __rebind_inp>::operator=()`.

4.848.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> void std::list<_Tp, _Alloc>::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Assigns a range to a list.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a list with copies of the elements in the range `[__first, __last)`.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 721 of file `stl_list.h`.

4.848.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::assign ( initializer_list<value_type> & __l ) [inline]`

Assigns an `initializer_list` to a list.

#### Parameters

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |
|------------------|---------------------------------------------------------------|

Replace the contents of the list with copies of the elements in the `initializer_list __l`. This is linear in `__l.size()`.

Definition at line 743 of file `stl_list.h`.

Referenced by `std::list<__inp, __rebind_inp>::assign()`.

4.848.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the list.

Definition at line 943 of file `stl_list.h`.

**4.848.3.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list<_Tp, _Alloc>::back ( )  
const [inline]`

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 955 of file stl\_list.h.

**4.848.3.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::begin ( )  
[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 758 of file stl\_list.h.

Referenced by `std::list<__inp, __rebind_inp>::crend()`, `std::list<__inp, __rebind_inp>::front()`, `std::list<__inp, __rebind_inp>::list()`, `std::list<_Tp, _Alloc>::merge()`, `std::list<_Tp, _Alloc>::operator=()`, `std::operator==()`, `std::list<__inp, __rebind_inp>::pop_front()`, `std::list<__inp, __rebind_inp>::push_front()`, `std::list<__inp, __rebind_inp>::rend()`, `std::list<__inp, __rebind_inp>::size()`, `std::list<_Tp, _Alloc>::sort()`, and `std::list<__inp, __rebind_inp>::splice()`.

**4.848.3.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::begin ( )  
const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 767 of file stl\_list.h.

**4.848.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::cbegin ( )  
const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 831 of file stl\_list.h.

**4.848.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::cend ( )  
const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 840 of file stl\_list.h.

**4.848.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::clear ( )  
[inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1228 of file stl\_list.h.

Referenced by `std::list<__inp, __rebind_inp>::operator=()`.

**4.848.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::rbegin ( )  
const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 849 of file stl\_list.h.

**4.848.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 858 of file `stl_list.h`.

**4.848.3.14** `template<typename _Tp, typename _Alloc > template<typename... _Args> list< _Tp, _Alloc >::iterator list::emplace ( iterator __position, _Args &&... __args )`

Constructs object in list before specified iterator.

#### Parameters

|                         |                                 |
|-------------------------|---------------------------------|
| <code>__position</code> | A const_iterator into the list. |
| <code>__args</code>     | Arguments.                      |

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args)...) before the specified location`. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 88 of file `list.tcc`.

Referenced by `std::list< __inp, __rebind_inp >::insert()`.

**4.848.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::list< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]`

Returns true if the list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 868 of file `stl_list.h`.

Referenced by `std::list< _Tp, _Alloc >::sort()`, and `std::list< __inp, __rebind_inp >::splice()`.

**4.848.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::end ( ) [inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 776 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::back()`, `std::list< __inp, __rebind_inp >::cbegin()`, `std::list< __inp, __rebind_inp >::list()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::operator=()`, `std::operator==( )`, `std::list< __inp, __rebind_inp >::push_back()`, `std::list< __inp, __rebind_inp >::rbegin()`, `std::list< __inp, __rebind_inp >::size()`, and `std::list< __inp, __rebind_inp >::splice()`.

**4.848.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list< _Tp, _Alloc >::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 785 of file `stl_list.h`.

**4.848.3.18** `template<typename _Tp, typename _Alloc > list< _Tp, _Alloc >::iterator list::erase ( iterator __position )`

Remove element at given position.

## Parameters

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

## Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 109 of file list.tcc.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::erase().

**4.848.3.19** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::erase ( iterator __first, iterator __last ) [inline]`

Remove a range of elements.

## Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

## Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1193 of file stl\_list.h.

**4.848.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list< _Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the list.

Definition at line 927 of file stl\_list.h.

**4.848.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list< _Tp, _Alloc >::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 935 of file stl\_list.h.

**4.848.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::list< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]`

Get a copy of the memory allocation object.

Definition at line 749 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert().

**4.848.3.23** `template<typename _Tp, typename _Alloc> list< _Tp, _Alloc >::iterator list::insert ( iterator __position, const value_type & __x )`

Inserts given value into list before specified iterator.

#### Parameters

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__x</code>        | Data to be inserted.       |

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 99 of file list.tcc.

**4.848.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into list before specified iterator.

#### Parameters

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__x</code>        | Data to be inserted.       |

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1089 of file stl\_list.h.

**4.848.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::insert ( iterator __p, initializer_list< value_type > __l ) [inline]`

Inserts the contents of an initializer\_list into list before specified iterator.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__p</code> | An iterator into the list.         |
| <code>__l</code> | An initializer_list of value_type. |

This function will insert copies of the data in the initializer\_list / into the list before the location specified by *p*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1106 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert().

4.848.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::insert ( iterator __position, size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the list.

#### Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the list.         |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1123 of file `stl_list.h`.

4.848.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> void std::list< _Tp, _Alloc >::insert ( iterator __position, _InputIterator __first, _InputIterator __last ) [inline]`

Inserts a range into the list.

#### Parameters

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__first</code>    | An input iterator.         |
| <code>__last</code>     | An input iterator.         |

This function will insert copies of the data in the range *[first,last)* into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1149 of file `stl_list.h`.

4.848.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::max_size ( ) const [inline],[noexcept]`

Returns the `size()` of the largest possible list.

Definition at line 878 of file `stl_list.h`.

4.848.3.29 `template<typename _Tp, typename _Alloc > void list::merge ( list< _Tp, _Alloc > && __x )`

Merge sorted lists.

#### Parameters

|                  |                       |
|------------------|-----------------------|
| <code>__x</code> | Sorted list to merge. |
|------------------|-----------------------|

Assumes that both `__x` and this list are sorted according to `operator<()`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

Definition at line 288 of file `list.tcc`.

References `std::begin()`, `std::list< _Tp, _Alloc >::begin()`, `std::end()`, and `std::list< _Tp, _Alloc >::end()`.

Referenced by `std::list< _Tp, _Alloc >::sort()`.

4.848.3.30 `template<typename _Tp, typename _Alloc > template<typename _StrictWeakOrdering > void list::merge ( list< _Tp, _Alloc > && __x, _StrictWeakOrdering __comp )`

Merge sorted lists according to comparison function.

#### Template Parameters

|                                  |                                          |
|----------------------------------|------------------------------------------|
| <code>_StrictWeakOrdering</code> | Comparison function defining sort order. |
|----------------------------------|------------------------------------------|

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__x</code>    | Sorted list to merge. |
| <code>__comp</code> | Comparison functor.   |

Assumes that both `__x` and this list are sorted according to `StrictWeakOrdering`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to `StrictWeakOrdering()`.

Definition at line 322 of file `list.tcc`.

References `std::begin()`, `std::list< _Tp, _Alloc >::begin()`, `std::end()`, and `std::list< _Tp, _Alloc >::end()`.

4.848.3.31 `template<typename _Tp, typename _Alloc > list< _Tp, _Alloc > & list::operator= ( const list< _Tp, _Alloc > & __x )`

List assignment operator.

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 185 of file `list.tcc`.

References `std::begin()`, `std::list< _Tp, _Alloc >::begin()`, `std::end()`, and `std::list< _Tp, _Alloc >::end()`.

4.848.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list< _Tp, _Alloc >::operator= ( list< _Tp, _Alloc > && __x ) [inline]`

List move assignment operator.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The contents of `__x` are moved into this list (without copying). `__x` is a valid, but unspecified list

Definition at line 667 of file `stl_list.h`.

4.848.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list< _Tp, _Alloc >::operator= ( initializer_list< value_type > __l ) [inline]`

List initializer list assignment operator.

## Parameters

|      |                                    |
|------|------------------------------------|
| ___/ | An initializer_list of value_type. |
|------|------------------------------------|

Replace the contents of the list with copies of the elements in the initializer\_list \_\_\_/. This is linear in l.size().

Definition at line 684 of file stl\_list.h.

**4.848.3.34** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::pop_back ( )`  
`[inline]`

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop\_back() is called.

Definition at line 1041 of file stl\_list.h.

**4.848.3.35** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::pop_front ( )`  
`[inline]`

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop\_front() is called.

Definition at line 1001 of file stl\_list.h.

**4.848.3.36** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::push_back ( const`  
`value_type & __x ) [inline]`

Add data to the end of the list.

## Parameters

|      |                   |
|------|-------------------|
| ___x | Data to be added. |
|------|-------------------|

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1015 of file stl\_list.h.

**4.848.3.37** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::push_front ( const`  
`value_type & __x ) [inline]`

Add data to the front of the list.

## Parameters

|      |                   |
|------|-------------------|
| ___x | Data to be added. |
|------|-------------------|

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 974 of file stl\_list.h.

**4.848.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list<_Tp, _Alloc>::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 794 of file `stl_list.h`.

**4.848.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 803 of file `stl_list.h`.

**4.848.3.40** `template<typename _Tp, typename _Alloc> void list::remove ( const _Tp & __value )`

Remove all elements equal to `value`.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__value</code> | The value to remove. |
|----------------------|----------------------|

Removes every element in the list equal to `value`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 239 of file `list.tcc`.

References `std::__addressof()`, `std::begin()`, and `std::end()`.

**4.848.3.41** `template<typename _Tp, typename _Alloc> template<typename _Predicate> void list::remove_if ( _Predicate __pred )`

Remove all elements satisfying a predicate.

#### Template Parameters

|                         |                                     |
|-------------------------|-------------------------------------|
| <code>_Predicate</code> | Unary predicate function or object. |
|-------------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 392 of file `list.tcc`.

References `std::begin()`, and `std::end()`.

**4.848.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list<_Tp, _Alloc>::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 812 of file `stl_list.h`.

**4.848.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 821 of file stl\_list.h.

**4.848.3.44** `template<typename _Tp, typename _Alloc > void list::resize ( size_type __new_size )`

Resizes the list to the specified number of elements.

#### Parameters

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain. |
|-------------------------|---------------------------------------------|

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 139 of file list.tcc.

References `std::begin()`, and `std::end()`.

**4.848.3.45** `template<typename _Tp, typename _Alloc > void list::resize ( size_type __new_size, const value_type & __x )`

Resizes the list to the specified number of elements.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain.       |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 154 of file list.tcc.

References `std::begin()`, and `std::end()`.

**4.848.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::reverse ( )`  
`[inline], [noexcept]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1449 of file stl\_list.h.

**4.848.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::size ( ) const`  
`[inline], [noexcept]`

Returns the number of elements in the list.

Definition at line 873 of file stl\_list.h.

**4.848.3.48** `template<typename _Tp, typename _Alloc > void list::sort ( )`

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 354 of file list.tcc.

References `std::begin()`, `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

4.848.3.49 `template<typename _Tp, typename _Alloc > template<typename _StrictWeakOrdering > void list::sort ( _StrictWeakOrdering __comp )`

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 431 of file list.tcc.

References `std::begin()`, `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

4.848.3.50 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice ( iterator __position, list< _Tp, _Alloc > && __x ) [inline]`

Insert contents of another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this != `__x`.

Definition at line 1248 of file stl\_list.h.

Referenced by `std::list< __inp, __rebind_inp >::insert()`, and `std::list< _Tp, _Alloc >::sort()`.

4.848.3.51 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice ( iterator __position, list< _Tp, _Alloc > && __x, iterator __i ) [inline]`

Insert element from another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__i</code>        | Iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1278 of file stl\_list.h.

4.848.3.52 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice ( iterator __position, list< _Tp, _Alloc > && __x, iterator __first, iterator __last ) [inline]`

Insert range from another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__first</code>    | Iterator referencing the start of range in x.      |
| <code>__last</code>     | Iterator referencing the end of range in x.        |

Removes elements in the range `[__first,__last)` and inserts them before `__position` in constant time.

Undefined if `__position` is in `[__first, __last)`.

Definition at line 1314 of file `stl_list.h`.

**4.848.3.53** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::swap ( list<_Tp, _Alloc> & _x ) [inline]`

Swaps data with another list.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A list of the same element and allocator types. |
|------------------|-------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 1210 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::operator=()`, `std::list< _Tp, _Alloc >::sort()`, and `std::swap()`.

**4.848.3.54** `template<typename _Tp, typename _Alloc > void list::unique ( )`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 267 of file `list.tcc`.

References `std::begin()`, and `std::end()`.

**4.848.3.55** `template<typename _Tp, typename _Alloc > template<typename _BinaryPredicate > void list::unique ( _BinaryPredicate __binary_pred )`

Remove consecutive elements satisfying a predicate.

#### Template Parameters

|                                |                                      |
|--------------------------------|--------------------------------------|
| <code>__BinaryPredicate</code> | Binary predicate function or object. |
|--------------------------------|--------------------------------------|

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 410 of file `list.tcc`.

References `std::begin()`, and `std::end()`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 4.849 std::locale Class Reference

### Classes

- class [facet](#)

*Localization functionality base class.*

*The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.*

- class [id](#)

*Facet ID class.*

*The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.*

## Public Types

- typedef int [category](#)

## Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &\_\_other) throw ()
- [locale](#) (const char \*\_\_s)
- [locale](#) (const [locale](#) &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [locale](#) &\_\_add, [category](#) \_\_cat)
- template<typename \_Facet >  
[locale](#) (const [locale](#) &\_\_other, \_Facet \*\_\_f)
- ~[locale](#) () throw ()
- template<typename \_Facet >  
[locale](#) combine (const [locale](#) &\_\_other) const
- [string](#) name () const
- bool [operator!=](#) (const [locale](#) &\_\_other) const throw ()
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
bool [operator\(\)](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s2) const
- template<typename \_Char, typename \_Traits, typename \_Alloc >  
bool [operator\(\)](#) (const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &\_\_other) throw ()
- bool [operator==](#) (const [locale](#) &\_\_other) const throw ()

## Static Public Member Functions

- static const [locale](#) & [classic](#) ()
- static [locale](#) [global](#) (const [locale](#) &\_\_loc)

## Static Public Attributes

- static const [category](#) [none](#)
- static const [category](#) [ctype](#)
- static const [category](#) [numeric](#)
- static const [category](#) [collate](#)
- static const [category](#) [time](#)
- static const [category](#) [monetary](#)
- static const [category](#) [messages](#)
- static const [category](#) [all](#)

## Friends

- template<typename \_Cache >  
struct **\_\_use\_cache**
- class **\_Impl**
- class **facet**
- template<typename \_Facet >  
bool **has\_facet** (const **locale** &) throw ()
- template<typename \_Facet >  
const \_Facet & **use\_facet** (const **locale** &)

## 4.849.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file locale\_classes.h.

## 4.849.2 Member Typedef Documentation

## 4.849.2.1 typedef int std::locale::category

Definition of locale::category.

Definition at line 67 of file locale\_classes.h.

## 4.849.3 Constructor &amp; Destructor Documentation

## 4.849.3.1 std::locale::locale ( ) throw ()

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by combine().

## 4.849.3.2 std::locale::locale ( const locale &amp; \_\_other ) throw ()

Copy constructor.

Constructs a copy of *other*.

## Parameters

|                |                     |
|----------------|---------------------|
| <b>__other</b> | The locale to copy. |
|----------------|---------------------|

## 4.849.3.3 std::locale::locale ( const char \* \_\_s ) [explicit]

Named locale constructor.

Constructs a copy of the named C library locale.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

#### Exceptions

|                                 |                                                     |
|---------------------------------|-----------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is null or an undefined locale. |
|---------------------------------|-----------------------------------------------------|

#### 4.849.3.4 std::locale::locale ( const locale & \_\_base, const char \* \_\_s, category \_\_cat )

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

#### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

#### Exceptions

|                                 |                                                     |
|---------------------------------|-----------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is null or an undefined locale. |
|---------------------------------|-----------------------------------------------------|

#### 4.849.3.5 std::locale::locale ( const locale & \_\_base, const locale & \_\_add, category \_\_cat )

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                            |
| <code>__add</code>  | The locale to use facets from.                                 |
| <code>__cat</code>  | Set of categories defining the facets to use from <i>add</i> . |

#### 4.849.3.6 template<typename \_Facet > std::locale::locale ( const locale & \_\_other, \_Facet \* \_\_f )

Construct locale with another facet.

Constructs a copy of the locale `__other`. The facet `__f` is added to `__other`, replacing an existing facet of type `Facet` if there is one. If `__f` is null, this locale is a copy of `__other`.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__other</code> | The locale to copy.  |
| <code>__f</code>     | The facet to add in. |

Definition at line 45 of file `locale_classes.tcc`.

## 4.849.3.7 std::locale::~~locale ( ) throw ()

Locale destructor.

## 4.849.4 Member Function Documentation

## 4.849.4.1 static const locale&amp; std::locale::classic ( ) [static]

Return reference to the C locale.

## 4.849.4.2 template&lt;typename \_Facet &gt; locale std::locale::combine ( const locale &amp; \_\_other ) const

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type *Facet* from the locale *other* into the new locale.

## Template Parameters

|               |                                   |
|---------------|-----------------------------------|
| <i>_Facet</i> | The facet type to copy from other |
|---------------|-----------------------------------|

## Parameters

|                |                          |
|----------------|--------------------------|
| <i>__other</i> | The locale to copy from. |
|----------------|--------------------------|

## Returns

Newly constructed locale.

## Exceptions

|                           |                                                        |
|---------------------------|--------------------------------------------------------|
| <i>std::runtime_error</i> | if <i>__other</i> has no facet of type <i>_Facet</i> . |
|---------------------------|--------------------------------------------------------|

Definition at line 63 of file locale\_classes.tcc.

References locale().

## 4.849.4.3 static locale std::locale::global ( const locale &amp; \_\_loc ) [static]

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC\_ALL, loc.name()).

## Parameters

|              |                                |
|--------------|--------------------------------|
| <i>__loc</i> | The new locale to make global. |
|--------------|--------------------------------|

## Returns

Copy of the old global locale.

## 4.849.4.4 string std::locale::name ( ) const

Return locale name.

**Returns**

Locale name or "\*" if unnamed.

**4.849.4.5** `bool std::locale::operator!=( const locale & __other ) const throw () [inline]`

Locale inequality.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__other</code> | The locale to compare against. |
|----------------------|--------------------------------|

**Returns**

`! (*this == __other)`

Definition at line 235 of file locale\_classes.h.

References `operator==()`.

**4.849.4.6** `template<typename _Char , typename _Traits , typename _Alloc > bool std::locale::operator() ( const basic_string< _Char, _Traits, _Alloc > & __s1, const basic_string< _Char, _Traits, _Alloc > & __s2 ) const`

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector `v` of strings could be sorted according to locale `loc` by doing:

```
std::sort(v.begin(), v.end(), loc);
```

**Parameters**

|                   |                           |
|-------------------|---------------------------|
| <code>__s1</code> | First string to compare.  |
| <code>__s2</code> | Second string to compare. |

**Returns**

True if `collate<_Char>` facet compares `__s1 < __s2`, else false.

**4.849.4.7** `const locale& std::locale::operator= ( const locale & __other ) throw ()`

Assignment operator.

Set this locale to be a copy of *other*.

**Parameters**

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

**Returns**

A reference to this locale.

**4.849.4.8** `bool std::locale::operator==( const locale & __other ) const throw ()`

Locale equality.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__other</code> | The locale to compare against. |
|----------------------|--------------------------------|

**Returns**

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

Referenced by operator!=().

**4.849.5 Friends And Related Function Documentation**

**4.849.5.1** `template<typename _Facet > bool has_facet ( const locale & ) throw ()` `[friend]`

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Template Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

**Returns**

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file locale\_classes.tcc.

**4.849.5.2** `template<typename _Facet > const _Facet& use_facet ( const locale & )` `[friend]`

Return a facet.

use\_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has\_facet(locale) is true, there is a suitable facet to return. It throws std::bad\_cast if the locale doesn't contain a facet of type Facet.

**Template Parameters**

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

## Returns

Reference to facet of type Facet.

## Exceptions

|                            |                                                                             |
|----------------------------|-----------------------------------------------------------------------------|
| <code>std::bad_cast</code> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|----------------------------|-----------------------------------------------------------------------------|

Definition at line 132 of file `locale_classes.tcc`.

## 4.849.6 Member Data Documentation

4.849.6.1 `const category std::locale::all` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

4.849.6.2 `const category std::locale::collate` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

4.849.6.3 `const category std::locale::ctype` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

4.849.6.4 `const category std::locale::messages` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

#### 4.849.6.5 `const category std::locale::monetary` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

#### 4.849.6.6 `const category std::locale::none` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

#### 4.849.6.7 `const category std::locale::numeric` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

#### 4.849.6.8 `const category std::locale::time` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

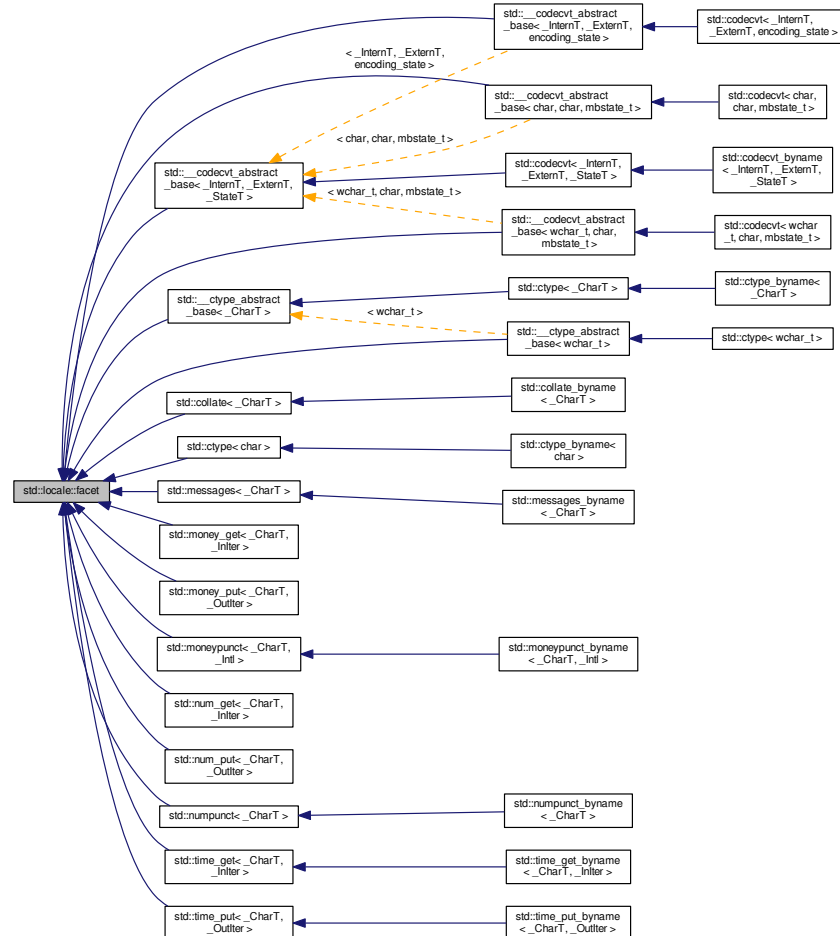
Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 4.850 std::locale::facet Class Reference

Inheritance diagram for std::locale::facet:



## Protected Member Functions

- [facet](#) (size\_t \_\_refs=0) throw ()
- virtual [~facet](#) ()

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Friends

- class **locale**
- class **locale::\_Impl**

## 4.850.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

Definition at line 338 of file `locale_classes.h`.

## 4.850.2 Constructor &amp; Destructor Documentation

4.850.2.1 `std::locale::facet::facet ( size_t __refs = 0 ) throw ()` `[inline]`, `[explicit]`, `[protected]`

Facet constructor.

This is the constructor provided by the standard. If `refs` is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

## Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__refs</code> | The initial value for reference count. |
|---------------------|----------------------------------------|

Definition at line 370 of file `locale_classes.h`.

4.850.2.2 `virtual std::locale::facet::~~facet ( )` `[protected]`, `[virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

4.851 `std::locale::id` Class Reference

## Public Member Functions

- `id ()`
- `size_t _M_id () const throw ()`

## Friends

- `template<typename _Facet >`  
`bool has_facet (const locale &) throw ()`
- class **locale**
- class **locale::\_Impl**
- `template<typename _Facet >`  
`const _Facet & use_facet (const locale &)`

## 4.851.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Definition at line 436 of file `locale_classes.h`.

## 4.851.2 Constructor &amp; Destructor Documentation

## 4.851.2.1 std::locale::id::id( ) [inline]

Constructor.

Definition at line 467 of file `locale_classes.h`.

## 4.851.3 Friends And Related Function Documentation

## 4.851.3.1 template&lt;typename \_Facet &gt; bool has\_facet( const locale &amp; ) throw() [friend]

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

## Template Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

## Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

## Returns

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file `locale_classes.tcc`.

## 4.851.3.2 template&lt;typename \_Facet &gt; const \_Facet&amp; use\_facet( const locale &amp; ) [friend]

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

## Template Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

## Returns

Reference to facet of type `Facet`.

## Exceptions

|                            |                                                                             |
|----------------------------|-----------------------------------------------------------------------------|
| <code>std::bad_cast</code> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|----------------------------|-----------------------------------------------------------------------------|

Definition at line 132 of file `locale_classes.tcc`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

4.852 `std::lock_guard< _Mutex >` Class Template Reference

## Public Types

- typedef `_Mutex` **`mutex_type`**

## Public Member Functions

- **`lock_guard`** (`mutex_type` &`_m`)
- **`lock_guard`** (`mutex_type` &`_m`, [adopt\\_lock\\_t](#))
- **`lock_guard`** (const [lock\\_guard](#) &)=delete
- **`lock_guard` & operator=** (const [lock\\_guard](#) &)=delete

## 4.852.1 Detailed Description

```
template<typename _Mutex>class std::lock_guard< _Mutex >
```

Scoped lock idiom.

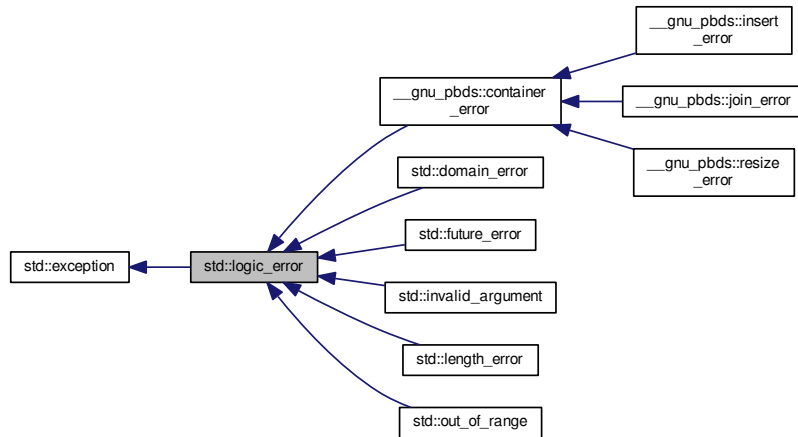
Definition at line 406 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 4.853 std::logic\_error Class Reference

Inheritance diagram for std::logic\_error:



## Public Member Functions

- [logic\\_error](#) (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

## 4.853.1 Detailed Description

One of two subclasses of exception.

Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 55 of file `stdexcept`.

## 4.853.2 Constructor &amp; Destructor Documentation

## 4.853.2.1 std::logic\_error::logic\_error ( const string &amp; \_\_arg ) [explicit]

Takes a character string describing the error.

## 4.853.3 Member Function Documentation

## 4.853.3.1 virtual const char\* std::logic\_error::what ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

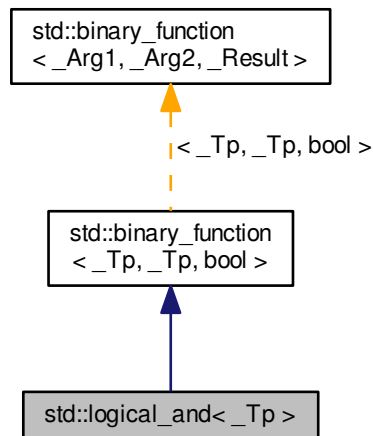
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

#### 4.854 std::logical\_and< \_Tp > Struct Template Reference

Inheritance diagram for std::logical\_and< \_Tp >:



##### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

##### Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

##### 4.854.1 Detailed Description

template<typename `_Tp`> struct std::logical\_and< `_Tp` >

One of the [Boolean operations functors](#).

Definition at line 268 of file `stl_function.h`.

## 4.854.2 Member Typedef Documentation

## 4.854.2.1 typedef \_Tp std::binary\_function&lt; \_Tp, \_Tp, bool &gt;::first\_argument\_type [inherited]

first\_argument\_type is the type of the first argument

Definition at line 117 of file stl\_function.h.

## 4.854.2.2 typedef bool std::binary\_function&lt; \_Tp, \_Tp, bool &gt;::result\_type [inherited]

result\_type is the return type

Definition at line 123 of file stl\_function.h.

## 4.854.2.3 typedef \_Tp std::binary\_function&lt; \_Tp, \_Tp, bool &gt;::second\_argument\_type [inherited]

second\_argument\_type is the type of the second argument

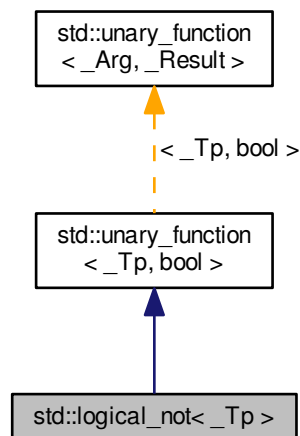
Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.855 std::logical\_not&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::logical\_not< \_Tp >:



## Public Types

- typedef \_Tp [argument\\_type](#)
- typedef bool [result\\_type](#)

#### Public Member Functions

- `bool operator() (const _Tp &__x) const`

##### 4.855.1 Detailed Description

`template<typename _Tp>struct std::logical_not<_Tp>`

One of the [Boolean operations functors](#).

Definition at line 286 of file `stl_function.h`.

##### 4.855.2 Member Typedef Documentation

4.855.2.1 `typedef _Tp std::unary_function<_Tp, bool>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.855.2.2 `typedef bool std::unary_function<_Tp, bool>::result_type` `[inherited]`

`result_type` is the return type

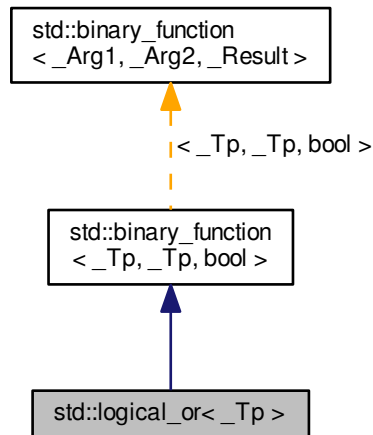
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.856 std::logical\_or&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::logical\_or< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

## 4.856.1 Detailed Description

```
template<typename _Tp>struct std::logical_or< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 277 of file `stl_function.h`.

## 4.856.2 Member Typedef Documentation

4.856.2.1 typedef `_Tp` `std::binary_function< _Tp, _Tp, bool >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.856.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.856.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.857 `std::lognormal_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- `typedef _RealType` [result\\_type](#)

### Public Member Functions

- `lognormal_distribution` (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- `lognormal_distribution` (`const` [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &\_\_p)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` (`result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `_RealType m` () `const`
- `result\_type max` () `const`
- `result\_type min` () `const`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator()` (`_UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &\_\_p)
- `param\_type param` () `const`
- `void param` (`const` [param\\_type](#) &\_\_param)
- `void reset` ()
- `_RealType s` () `const`

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT,  
\_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::lognormal\_distribution< \_RealType1 > &\_\_x)
- bool operator== (const lognormal\_distribution &\_\_d1, const lognormal\_distribution &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
std::basic\_istream< \_CharT,  
\_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::lognormal\_distribution< \_RealType1 > &\_\_x)

## 4.857.1 Detailed Description

template<typename \_RealType = double>class std::lognormal\_distribution< \_RealType >

A lognormal\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m,s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2298 of file random.h.

## 4.857.2 Member Typedef Documentation

4.857.2.1 template<typename \_RealType = double> typedef \_RealType std::lognormal\_distribution< \_RealType >::result\_type

The type of the range of the distribution.

Definition at line 2301 of file random.h.

## 4.857.3 Member Function Documentation

4.857.3.1 template<typename \_RealType = double> result\_type std::lognormal\_distribution< \_RealType >::max ( ) const  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 2389 of file random.h.

References std::numeric\_limits<\_Tp>::max().

4.857.3.2 template<typename \_RealType = double> result\_type std::lognormal\_distribution< \_RealType >::min ( ) const  
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 2382 of file random.h.

4.857.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type  
std::lognormal_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng )  
[inline]`

Generating functions.

Definition at line 2397 of file random.h.

References `std::lognormal_distribution<_RealType>::operator()()`.

Referenced by `std::lognormal_distribution<_RealType>::operator()()`.

4.857.3.4 `template<typename _RealType = double> param_type std::lognormal_distribution<_RealType>::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 2367 of file random.h.

4.857.3.5 `template<typename _RealType = double> void std::lognormal_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2375 of file random.h.

4.857.3.6 `template<typename _RealType = double> void std::lognormal_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 2349 of file random.h.

References `std::normal_distribution<_RealType>::reset()`.

#### 4.857.4 Friends And Related Function Documentation

4.857.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const  
std::lognormal_distribution<_RealType1> & __x ) [friend]`

Inserts a `lognormal_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                 |
| <code>__x</code>  | A <code>lognormal_distribution</code> random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

4.857.4.2 `template<typename _RealType = double> bool operator==( const lognormal_distribution<_RealType> & __d1, const lognormal_distribution<_RealType> & __d2 ) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2434 of file `random.h`.

4.857.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::lognormal_distribution<_RealType1> & __x ) [friend]`

Extracts a `lognormal_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                      |
| <code>__x</code>  | A <code>lognormal_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.858 `std::lognormal_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `lognormal_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- `_RealType m` () const
- `_RealType s` () const

## Friends

- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.858.1 Detailed Description

```
template<typename _RealType = double>struct std::lognormal_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 2307 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.859 `std::make_signed< _Tp >` Struct Template Reference

## Public Types

- typedef `__make_signed_selector< _Tp >::__type` **type**

## 4.859.1 Detailed Description

```
template<typename _Tp>struct std::make_signed< _Tp >
```

`make_signed`

Definition at line 1612 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.860 `std::make_unsigned< _Tp >` Struct Template Reference

## Public Types

- typedef `__make_unsigned_selector< _Tp >::__type` **type**

## 4.860.1 Detailed Description

```
template<typename _Tp>struct std::make_unsigned< _Tp >
```

`make_unsigned`

Definition at line 1530 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.861 `std::map< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Pair_alloc_type::const_pointer` **const\_pointer**
- typedef `_Pair_alloc_type::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Pair_alloc_type::pointer` **pointer**
- typedef `_Pair_alloc_type::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- `map` ()
- `map` (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `map` (const `map` &\_\_x)
- `map` (`map` &&\_\_x) noexcept(is\_nothrow\_copy\_constructible< `_Compare` >)
- `map` (initializer\_list< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- template<typename `_InputIterator` >  
`map` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` >  
`map` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `mapped_type` & `at` (const `key_type` &\_\_k)
- const `mapped_type` & `at` (const `key_type` &\_\_k) const
- iterator `begin` () noexcept
- const\_iterator `begin` () const noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- size\_type `count` (const `key_type` &\_\_x) const
- const\_reverse\_iterator `crbegin` () const noexcept
- const\_reverse\_iterator `crend` () const noexcept
- template<typename... `_Args`>  
`std::pair`< iterator, bool > `emplace` (`_Args` &&...\_\_args)
- template<typename... `_Args`>  
iterator `emplace_hint` (const\_iterator \_\_pos, `_Args` &&...\_\_args)

- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- **std::pair**< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- **std::pair**< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- iterator **erase** (const\_iterator \_\_position)
- iterator **erase** (iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- allocator\_type **get\_allocator** () const noexcept
- **std::pair**< iterator, bool > **insert** (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
**std::pair**< iterator, bool > **insert** (\_Pair &&\_\_x)
- void **insert** (std::initializer\_list< value\_type > \_\_list)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- key\_compare **key\_comp** () const
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- size\_type **max\_size** () const noexcept
- map & **operator=** (const map &\_\_x)
- map & **operator=** (map &&\_\_x)
- map & **operator=** (initializer\_list< value\_type > \_\_l)
- mapped\_type & **operator[]** (const key\_type &\_\_k)
- mapped\_type & **operator[]** (key\_type &&\_\_k)
- **reverse\_iterator** **rbegin** () noexcept
- **const\_reverse\_iterator** **rbegin** () const noexcept
- **reverse\_iterator** **rend** () noexcept
- **const\_reverse\_iterator** **rend** () const noexcept
- size\_type **size** () const noexcept
- void **swap** (map &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- value\_compare **value\_comp** () const

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator<** (const map< \_K1, \_T1, \_C1, \_A1 > &, const map< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator==** (const map< \_K1, \_T1, \_C1, \_A1 > &, const map< \_K1, \_T1, \_C1, \_A1 > &)

## 4.861.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> class std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Template Parameters

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                |
| <code>_Tp</code>      | Type of mapped objects.                                                             |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .        |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 96 of file `stl_map.h`.

## 4.861.2 Constructor &amp; Destructor Documentation

```
4.861.2.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map () [inline]
```

Default constructor creates no elements.

Definition at line 160 of file `stl_map.h`.

```
4.861.2.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (const _Compare &
__comp, const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a map with no elements.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

Definition at line 169 of file `stl_map.h`.

```
4.861.2.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (const map< _Key,
_Tp, _Compare, _Alloc > & __x) [inline]
```

Map copy constructor.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A map of identical element and allocator types. |
|------------------|-------------------------------------------------|

The newly-created map uses a copy of the allocation object used by \_\_x.

Definition at line 180 of file stl\_map.h.

```
4.861.2.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (map< _Key, _Tp,
_Compare, _Alloc > && __x) [inline], [noexcept]
```

Map move constructor.

#### Parameters

|     |                                                 |
|-----|-------------------------------------------------|
| __x | A map of identical element and allocator types. |
|-----|-------------------------------------------------|

The newly-created map contains the exact contents of \_\_x. The contents of \_\_x are a valid, but unspecified map.

Definition at line 191 of file stl\_map.h.

```
4.861.2.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (initializer_list<
value_type> & __l, const _Compare & __comp = _Compare(), const allocator_type & __a = allocator_type()
) [inline]
```

Builds a map from an initializer\_list.

#### Parameters

|        |                      |
|--------|----------------------|
| __l    | An initializer_list. |
| __comp | A comparison object. |
| __a    | An allocator object. |

Create a map consisting of copies of the elements in the initializer\_list \_\_l. This is linear in N if the range is already sorted, and NlogN otherwise (where N is \_\_l.size()).

Definition at line 206 of file stl\_map.h.

```
4.861.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > std::map< _Key, _Tp, _Compare,
_Alloc >::map (_InputIterator __first, _InputIterator __last) [inline]
```

Builds a map from a range.

#### Parameters

|         |                    |
|---------|--------------------|
| __first | An input iterator. |
| __last  | An input iterator. |

Create a map consisting of copies of the elements from [\_\_first,\_\_last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(\_\_first,\_\_last)).

Definition at line 224 of file stl\_map.h.

```
4.861.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > std::map< _Key, _Tp, _Compare,
_Alloc >::map (_InputIterator __first, _InputIterator __last, const _Compare & __comp, const allocator_type & __a =
allocator_type()) [inline]
```

Builds a map from a range.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 241 of file `stl_map.h`.

## 4.861.3 Member Function Documentation

4.861.3.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map<_Key, _Tp, _Compare, _Alloc>::at ( const key_type & __k ) [inline]`

Access to map data.

## Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

## Returns

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

## Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

Definition at line 501 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::end()`, `std::map<_Key, _Tp, _Compare, _Alloc>::key_comp()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound()`.

4.861.3.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin ( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 320 of file `stl_map.h`.

4.861.3.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 329 of file `stl_map.h`.

```
4.861.3.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::cbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 393 of file stl\_map.h.

```
4.861.3.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::cend ()
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 402 of file stl\_map.h.

```
4.861.3.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc>::clear ()
[inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 787 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator=().

```
4.861.3.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc>::count (const
key_type & __x) const [inline]
```

Finds the number of elements with given key.

#### Parameters

|     |                                          |
|-----|------------------------------------------|
| __x | Key of (key, value) pairs to be located. |
|-----|------------------------------------------|

#### Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 847 of file stl\_map.h.

```
4.861.3.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>
>::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 411 of file stl\_map.h.

```
4.861.3.9 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc
>::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 420 of file stl\_map.h.

```
4.861.3.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> std::pair<iterator, bool> std::map<
_Key, _Tp, _Compare, _Alloc >::emplace (_Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

#### Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 540 of file stl\_map.h.

```
4.861.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator std::map<_Key, _Tp,
_Compare, _Alloc >::emplace_hint (const_iterator __pos, _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

#### Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

#### Returns

An iterator that points to the element with key of the std::pair built from \_\_args (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 570 of file stl\_map.h.

```
4.861.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::map< _Key, _Tp, _Compare, _Alloc >::empty () const
[inline], [noexcept]
```

Returns true if the map is empty. (Thus begin() would equal end().)

Definition at line 429 of file stl\_map.h.

```
4.861.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::end ()
[inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 338 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

```
4.861.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::end ()
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 347 of file stl\_map.h.

```
4.861.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::map< _Key, _Tp, _Compare, _Alloc
>::equal_range (const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|     |                                          |
|-----|------------------------------------------|
| __x | Key of (key, value) pairs to be located. |
|-----|------------------------------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 916 of file stl\_map.h.

```
4.861.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator> std::map< _Key, _Tp,
_Compare, _Alloc >::equal_range (const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

## Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 935 of file `stl_map.h`.

```
4.861.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __position) [inline]
```

Erases an element from a map.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 690 of file `stl_map.h`.

```
4.861.3.18 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> size_type std::map<_Key, _Tp, _Compare, _Alloc >::erase (const
key_type & __x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

## Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 725 of file stl\_map.h.

```
4.861.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __first, const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a map.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

#### Returns

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 745 of file stl\_map.h.

```
4.861.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc >::find (const
key_type & __x) [inline]
```

Tries to locate an element in a map.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 820 of file stl\_map.h.

```
4.861.3.21 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::find (const
key_type & __x) const [inline]
```

Tries to locate an element in a map.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 835 of file `stl_map.h`.

```
4.861.3.22 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator
() const [inline],[noexcept]
```

Get a copy of the memory allocation object.

Definition at line 310 of file `stl_map.h`.

```
4.861.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::map< _Key, _Tp, _Compare, _Alloc
>::insert(const value_type & __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 594 of file `stl_map.h`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`.

```
4.861.3.24 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
std::initializer_list< value_type > __list) [inline]
```

Attempts to insert a list of `std::pairs` into the map.

**Parameters**

|                     |                                                                                 |
|---------------------|---------------------------------------------------------------------------------|
| <code>__list</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted. |
|---------------------|---------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 615 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

4.861.3.25 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x ) [inline]`

Attempts to insert a std::pair into the map.

#### Parameters

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>        | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 644 of file stl\_map.h.

4.861.3.26 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > void std::map< _Key, _Tp, _Compare, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]`

Template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 670 of file stl\_map.h.

4.861.3.27 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> key_compare std::map< _Key, _Tp, _Compare, _Alloc >::key_comp ( ) const [inline]`

Returns the key comparison object out of which the map was constructed.

Definition at line 796 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

4.861.3.28 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound ( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 862 of file stl\_map.h.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::at()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`.

```
4.861.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound
(const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 877 of file stl\_map.h.

```
4.861.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size ()
const [inline], [noexcept]
```

Returns the maximum size of the map.

Definition at line 439 of file stl\_map.h.

```
4.861.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (const
map< _Key, _Tp, _Compare, _Alloc > & __x) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A map of identical element and allocator types. |
|------------------|-------------------------------------------------|

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 264 of file stl\_map.h.

```
4.861.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (map<
_Key, _Tp, _Compare, _Alloc > && __x) [inline]
```

Map move assignment operator.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A map of identical element and allocator types. |
|------------------|-------------------------------------------------|

The contents of `__x` are moved into this map (without copying). `__x` is a valid, but unspecified map.

Definition at line 279 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::swap()`.

```
4.861.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
initializer_list< value_type > __l) [inline]
```

Map list assignment operator.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

This function fills a map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 300 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

```
4.861.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map< _Key, _Tp, _Compare, _Alloc >::operator[] (
const key_type & __k) [inline]
```

Subscript ( `[]` ) access to map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 456 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::end()`, `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`, `std::map< _Key, _Tp, _Compare, _Alloc >::key_comp()`, `std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound()`, and `std::piecewise_construct`.

```
4.861.3.35 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin
() [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 356 of file stl\_map.h.

```
4.861.3.36 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
>::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 365 of file stl\_map.h.

```
4.861.3.37 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend (
) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 374 of file stl\_map.h.

```
4.861.3.38 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
>::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 383 of file stl\_map.h.

```
4.861.3.39 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc >::size () const
[inline], [noexcept]
```

Returns the size of the map.

Definition at line 434 of file stl\_map.h.

```
4.861.3.40 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc >::swap (map< _Key,
_Tp, _Compare, _Alloc > & __x) [inline]
```

Swaps data with another map.

#### Parameters

|     |                                                |
|-----|------------------------------------------------|
| __x | A map of the same element and allocator types. |
|-----|------------------------------------------------|

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 777 of file stl\_map.h.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc>::operator=()`, and `std::swap()`.

4.861.3.41 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::upper_bound(  
const key_type & _x ) [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 887 of file `stl_map.h`.

4.861.3.42 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::upper_bound  
( const key_type & _x ) const [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or `end()`.

Definition at line 897 of file `stl_map.h`.

4.861.3.43 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> value_compare std::map<_Key, _Tp, _Compare, _Alloc>::value_comp  
( ) const [inline]`

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 804 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

## 4.862 `std::mask_array<_Tp>` Class Template Reference

### Public Types

- `typedef _Tp value_type`

### Public Member Functions

- `mask_array` (const `mask_array` &)

- void **operator%=(const valarray< \_Tp > &)** const
- template<class \_Dom >  
void **operator%=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator&=(const valarray< \_Tp > &)** const
- template<class \_Dom >  
void **operator&=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator\*=(const valarray< \_Tp > &)** const
- template<class \_Dom >  
void **operator\*=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator+=(const valarray< \_Tp > &)** const
- template<class \_Dom >  
void **operator+=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator-=(const valarray< \_Tp > &)** const
- template<class \_Dom >  
void **operator-=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator/=(const valarray< \_Tp > &)** const
- template<class \_Dom >  
void **operator/=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator<=<=** (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator<<=** (const \_Expr< \_Dom, \_Tp > &) const
- **mask\_array & operator=** (const **mask\_array** &)
- void **operator=** (const valarray< \_Tp > &) const
- void **operator=** (const \_Tp &) const
- template<class \_Dom >  
void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Ex >  
void **operator=** (const \_Expr< \_Ex, \_Tp > &\_\_e) const
- void **operator>>=** (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator>>=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator^=** (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator^=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator|=** (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator|=** (const \_Expr< \_Dom, \_Tp > &) const

## Friends

- class **valarray< \_Tp >**

### 4.862.1 Detailed Description

template<class \_Tp>class std::mask\_array< \_Tp >

Reference to selected subset of an array.

A mask\_array is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a mask\_array is to call operator[](valarray<bool>) on a valarray. The returned mask\_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if a mask\_array is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

## Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

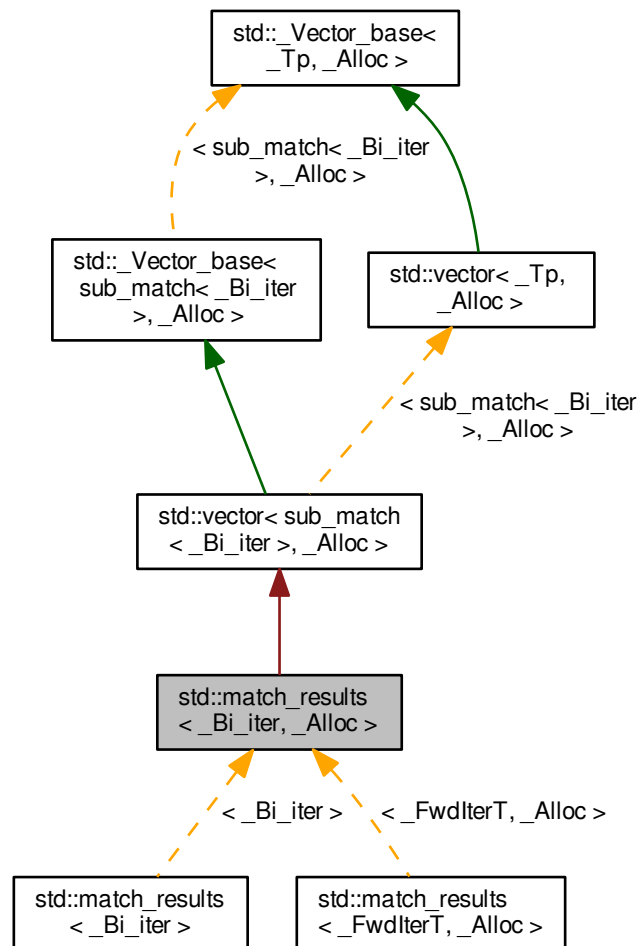
Definition at line 62 of file mask\_array.h.

The documentation for this class was generated from the following file:

- [mask\\_array.h](#)

## 4.863 std::match\_results&lt; \_Bi\_iter, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::match\_results< \_Bi\_iter, \_Alloc >:



## Public Member Functions

- `bool ready () const`

## Private Types

- `typedef`  
`_Alloc_traits::const_pointer` **`const_pointer`**
- `typedef` `std::reverse_iterator`  
`< const_iterator >` **`const_reverse_iterator`**
- `typedef` `_Base::pointer` **`pointer`**
- `typedef` `std::reverse_iterator`  
`< iterator >` **`reverse_iterator`**

## Private Member Functions

- `pointer _M_allocate (size_t __n)`
- `pointer _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)`
- `void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`
- `void _M_deallocate (pointer __p, size_t __n)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize (size_type __n)`
- `void _M_emplace_back_aux (_Args &&... __args)`
- `void _M_erase_at_end (pointer __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `void _M_initialize_dispatch (_Integer __n, _Integer __value, __true_type)`
- `void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_insert_aux (iterator __position, _Args &&... __args)`
- `void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `void _M_range_insert (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `bool _M_shrink_to_fit ()`
- `void assign (size_type __n, const value_type &__val)`
- `void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (initializer_list< value_type > __l)`
- `reference at (size_type __n)`
- `const_reference at (size_type __n) const`

- [reference back](#) ()
- [const\\_reference back](#) () const
- iterator [begin](#) () noexcept
- size\_type [capacity](#) () const noexcept
- void [clear](#) () noexcept
- [const\\_reverse\\_iterator crbegin](#) () const noexcept
- [const\\_reverse\\_iterator crend](#) () const noexcept
- [sub\\_match](#)<\_Bi\_iter > \* [data](#) () noexcept
- const [sub\\_match](#)<\_Bi\_iter > \* [data](#) () const noexcept
- iterator [emplace](#) (iterator \_\_position, \_Args &&... \_\_args)
- void [emplace\\_back](#) (\_Args &&... \_\_args)
- iterator [end](#) () noexcept
- iterator [erase](#) (iterator \_\_position)
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- [reference front](#) ()
- [const\\_reference front](#) () const
- iterator [insert](#) (iterator \_\_position, const [value\\_type](#) &\_\_x)
- iterator [insert](#) (iterator \_\_position, [value\\_type](#) &&\_\_x)
- void [insert](#) (iterator \_\_position, [initializer\\_list](#)< [value\\_type](#) > \_\_l)
- void [insert](#) (iterator \_\_position, size\_type \_\_n, const [value\\_type](#) &\_\_x)
- void [insert](#) (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [reference operator\[\]](#) (size\_type \_\_n)
- [const\\_reference operator\[\]](#) (size\_type \_\_n) const
- void [pop\\_back](#) ()
- void [push\\_back](#) (const [value\\_type](#) &\_\_x)
- void [push\\_back](#) ([value\\_type](#) &&\_\_x)
- [reverse\\_iterator rbegin](#) () noexcept
- [const\\_reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- void [reserve](#) (size\_type \_\_n)
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const [value\\_type](#) &\_\_x)
- void [shrink\\_to\\_fit](#) ()
- void [swap](#) ([vector](#) &\_\_x) noexcept([\\_Alloc\\_traits](#)

#### Private Attributes

- [\\_Vector\\_impl \\_M\\_impl](#)

#### Friends

- class [\\_\\_detail::SpecializedResults](#)<\_Bi\_iter, \_Alloc >

## 10.? Public Types

- typedef \_Alloc **allocator\_type**
- typedef [sub\\_match](#)<\_Bi\_iter > **value\_type**
- typedef const [value\\_type](#) & **const\_reference**
- typedef [const\\_reference](#) **reference**
- typedef \_Base\_type::const\_iterator **const\_iterator**
- typedef const\_iterator **iterator**
- typedef  
\_\_iter\_traits::difference\_type **difference\_type**
- typedef \_\_iter\_traits::value\_type **char\_type**
- typedef [allocator\\_traits](#)  
<\_Alloc >::size\_type **size\_type**
- typedef [std::basic\\_string](#)  
< char\_type > **string\_type**

## 28.10.1 Construction, Copying, and Destruction

- [match\\_results](#) (const \_Alloc &\_\_a=\_Alloc())
- [match\\_results](#) (const [match\\_results](#) &\_\_rhs)
- [match\\_results](#) ([match\\_results](#) &&\_\_rhs) noexcept
- [match\\_results](#) & operator= (const [match\\_results](#) &\_\_rhs)
- [match\\_results](#) & operator= ([match\\_results](#) &&\_\_rhs)
- [~match\\_results](#) ()

## 28.10.2 Size

- size\_type [size](#) () const
- size\_type [max\\_size](#) () const
- bool [empty](#) () const

## 10.3 Element Access

- difference\_type [length](#) (size\_type \_\_sub=0) const
- difference\_type [position](#) (size\_type \_\_sub=0) const
- [string\\_type](#) str (size\_type \_\_sub=0) const
- [const\\_reference](#) operator[] (size\_type \_\_sub) const
- [const\\_reference](#) prefix () const
- [const\\_reference](#) suffix () const
- const\_iterator [begin](#) () const
- const\_iterator [cbegin](#) () const
- const\_iterator [end](#) () const
- const\_iterator [cend](#) () const

### 10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter >`  
`_Out_iter format (_Out_iter __out, const char_type * __fmt_first, const char_type * __fmt_last, match_flag_type __flags=regex_constants::format_default) const`
- `template<typename _Out_iter, typename _St, typename _Sa >`  
`_Out_iter format (_Out_iter __out, const basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __flags=regex_constants::format_default) const`
- `template<typename _Out_iter, typename _St, typename _Sa >`  
`basic_string< char_type, _St, _Sa > format (const basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __flags=regex_constants::format_default) const`
- `string_type format (const char_type * __fmt, match_flag_type __flags=regex_constants::format_default) const`

### 10.5 Allocator

- `allocator_type get_allocator () const`

### 10.6 Swap

- `void swap (match_results & __that)`

#### 4.863.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> class std::match_results< _Bi_iter, _Alloc >
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_match` member `matched` is always true. The `sub_match` object stored at index `n` denotes what matched the marked sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters [`first`, `second`) which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of the sequence that was searched.

Definition at line 1429 of file `regex.h`.

#### 4.863.2 Constructor & Destructor Documentation

```
4.863.2.1 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter, _Alloc >::match_results (const _Alloc & __a = _Alloc()) [inline], [explicit]
```

Constructs a default `match_results` container.

**Postcondition**

size() returns 0 and str() returns an empty string.

Definition at line 1478 of file regex.h.

Referenced by std::match\_results<\_FwdIterT, \_Alloc>::operator=().

**4.863.2.2** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results ( const match_results<_Bi_iter, _Alloc> & __rhs ) [inline]`

Copy constructs a match\_results.

Definition at line 1485 of file regex.h.

**4.863.2.3** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results ( match_results<_Bi_iter, _Alloc> && __rhs ) [inline], [noexcept]`

Move constructs a match\_results.

Definition at line 1492 of file regex.h.

**4.863.2.4** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::~~match_results ( ) [inline]`

Destroys a match\_results object.

Definition at line 1519 of file regex.h.

**4.863.3 Member Function Documentation**

**4.863.3.1** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::begin ( ) const [inline]`

Gets an iterator to the start of the sub\_match collection.

Definition at line 1676 of file regex.h.

Referenced by std::operator==( ).

**4.863.3.2** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::cbegin ( ) const [inline]`

Gets an iterator to the start of the sub\_match collection.

Definition at line 1683 of file regex.h.

**4.863.3.3** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::cend ( ) const [inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1697 of file regex.h.

**4.863.3.4** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> bool std::match_results<_Bi_iter, _Alloc>::empty ( ) const [inline]`

Indicates if the match\_results contains no results.

## Return values

|              |                                        |
|--------------|----------------------------------------|
| <i>true</i>  | The match_results object is empty.     |
| <i>false</i> | The match_results object is not empty. |

Definition at line 1563 of file regex.h.

Referenced by std::match\_results< \_FwdIterT, \_Alloc >::end(), std::operator==( ), std::match\_results< \_FwdIterT, \_Alloc >::prefix(), and std::match\_results< \_FwdIterT, \_Alloc >::suffix().

4.863.3.5 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator  
std::match_results< _Bi_iter, _Alloc >::end ( ) const [inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1690 of file regex.h.

Referenced by std::match\_results< \_FwdIterT, \_Alloc >::cend(), and std::operator==( ).

4.863.3.6 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter  
> _Out_iter std::match_results< _Bi_iter, _Alloc >::format ( _Out_iter __out, const char_type * __fmt_first, const  
char_type * __fmt_last, match_flag_type __flags = regex_constants::format_default ) const [inline]`

## Precondition

ready() == true

**Todo** Implement this function.

Definition at line 1718 of file regex.h.

Referenced by std::match\_results< \_FwdIterT, \_Alloc >::format().

4.863.3.7 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter ,  
typename _St , typename _Sa > _Out_iter std::match_results< _Bi_iter, _Alloc >::format ( _Out_iter __out, const  
basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __flags = regex_constants::format_default ) const  
[inline]`

## Precondition

ready() == true

Definition at line 1728 of file regex.h.

4.863.3.8 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter ,  
typename _St , typename _Sa > basic_string<char_type, _St, _Sa> std::match_results< _Bi_iter, _Alloc >::format (   
const basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __flags = regex_constants::format_default )  
const [inline]`

## Precondition

ready() == true

Definition at line 1740 of file regex.h.

4.863.3.9 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> string_type  
std::match_results< _Bi_iter, _Alloc >::format ( const char_type * __fmt, match_flag_type __flags =  
regex_constants::format_default ) const [inline]`

**Precondition**

ready() == true

Definition at line 1752 of file regex.h.

**4.863.3.10** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> allocator_type  
std::match_results<_Bi_iter, _Alloc >::get_allocator ( ) const [inline]`

Gets a copy of the allocator.

Definition at line 1773 of file regex.h.

**4.863.3.11** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type  
std::match_results<_Bi_iter, _Alloc >::length ( size_type __sub = 0 ) const [inline]`

Gets the length of the indicated submatch.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

Definition at line 1582 of file regex.h.

**4.863.3.12** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> size_type std::match_results<  
_Bi_iter, _Alloc >::max_size ( ) const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

**Returns**

the number of matches found.

Definition at line 1554 of file regex.h.

**4.863.3.13** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results&  
std::match_results<_Bi_iter, _Alloc >::operator=( const match_results<_Bi_iter, _Alloc > & __rhs )  
[inline]`

Assigns rhs to \*this.

Definition at line 1500 of file regex.h.

**4.863.3.14** `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results&  
std::match_results<_Bi_iter, _Alloc >::operator=( match_results<_Bi_iter, _Alloc > && __rhs ) [inline]`

Move-assigns rhs to \*this.

Definition at line 1510 of file regex.h.

4.863.3.15 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_reference  
std::match_results<_Bi_iter, _Alloc>::operator[] ( size_type __sub ) const [inline]`

Gets a sub\_match reference for the match or submatch.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

#### Precondition

`ready() == true`

This function gets a reference to the indicated submatch, or the entire match if `__sub` is zero.

If `__sub >= size()` then this function returns a sub\_match with a special value indicating no submatch.

Definition at line 1630 of file regex.h.

4.863.3.16 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> difference_type  
std::match_results<_Bi_iter, _Alloc>::position ( size_type __sub = 0 ) const [inline]`

Gets the offset of the beginning of the indicated submatch.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

#### Precondition

`ready() == true`

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of `__sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if `__sub` is out of range.

Definition at line 1599 of file regex.h.

4.863.3.17 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_reference  
std::match_results<_Bi_iter, _Alloc>::prefix ( ) const [inline]`

Gets a sub\_match representing the match prefix.

#### Precondition

`ready() == true`

This function gets a reference to a sub\_match object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1647 of file regex.h.

Referenced by `std::operator==( )`, and `std::match_results<_FwdIterT, _Alloc>::position()`.

4.863.3.18 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> bool std::match_results<  
_Bi_iter, _Alloc>::ready ( ) const [inline]`

Indicates if the match\_results is ready.

## Return values

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>true</i>  | The object has a fully-established result state. |
| <i>false</i> | The object is not ready.                         |

Definition at line 1530 of file regex.h.

Referenced by std::operator==(), std::match\_results< \_FwdIterT, \_Alloc >::operator[](), std::match\_results< \_FwdIterT, \_Alloc >::prefix(), and std::match\_results< \_FwdIterT, \_Alloc >::suffix().

**4.863.3.19** template<typename *\_Bi\_iter*, typename *\_Alloc* = allocator<sub\_match<\_Bi\_iter> >> **size\_type** std::match\_results< *\_Bi\_iter*, *\_Alloc* >::size ( ) const [inline]

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark\_count() + 1 if a match was successful. Some matches may be empty.

## Returns

the number of matches found.

Definition at line 1547 of file regex.h.

Referenced by std::match\_results< \_FwdIterT, \_Alloc >::empty(), std::operator==(), std::match\_results< \_FwdIterT, \_Alloc >::operator[](), and std::match\_results< \_FwdIterT, \_Alloc >::position().

**4.863.3.20** template<typename *\_Bi\_iter*, typename *\_Alloc* = allocator<sub\_match<\_Bi\_iter> >> **string\_type** std::match\_results< *\_Bi\_iter*, *\_Alloc* >::str ( **size\_type** *\_\_sub* = 0 ) const [inline]

Gets the match or submatch converted to a string type.

## Parameters

|              |                         |
|--------------|-------------------------|
| <i>__sub</i> | indicates the submatch. |
|--------------|-------------------------|

## Precondition

ready() == true

This function gets the submatch (or match, if *\_\_sub* is zero) extracted from the target range and converted to the associated string type.

Definition at line 1615 of file regex.h.

**4.863.3.21** template<typename *\_Bi\_iter*, typename *\_Alloc* = allocator<sub\_match<\_Bi\_iter> >> **const\_reference** std::match\_results< *\_Bi\_iter*, *\_Alloc* >::suffix ( ) const [inline]

Gets a sub\_match representing the match suffix.

## Precondition

ready() == true

This function gets a reference to a sub\_match object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1664 of file regex.h.

Referenced by std::operator==().

4.863.3.22 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> void std::match_results<_Bi_iter, _Alloc >::swap ( match_results<_Bi_iter, _Alloc > & __that ) [inline]`

Swaps the contents of two match\_results.

Definition at line 1787 of file regex.h.

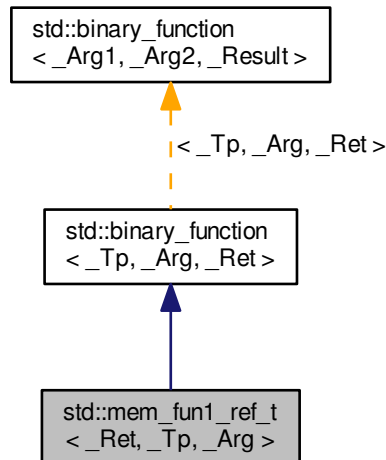
Referenced by std::swap().

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.864 std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

### Public Member Functions

- **mem\_fun1\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg))
- **\_Ret operator()** (\_Tp &\_\_r, \_Arg \_\_x) const

## 4.864.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`

One of the [adaptors for member pointers](#).

Definition at line 650 of file `stl_function.h`.

## 4.864.2 Member Typedef Documentation

4.864.2.1 `typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.864.2.2 `typedef _Ret std::binary_function< _Tp, _Arg, _Ret >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.864.2.3 `typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

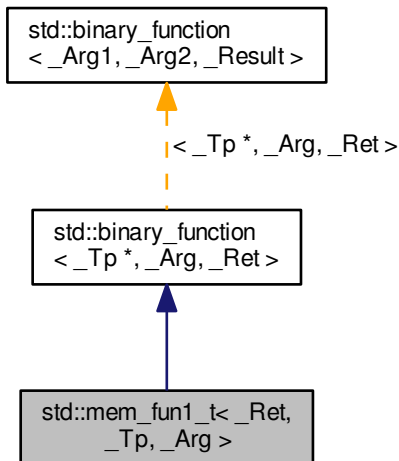
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.865 std::mem\_fun1\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef `_Tp` \* [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- **mem\_fun1\_t** (`_Ret` (`_Tp`::\* `__pf`) (`_Arg`))
- `_Ret` **operator()** (`_Tp` \* `__p`, `_Arg` `__x`) const

## 4.865.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg> class std::mem_fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 614 of file `stl_function.h`.

## 4.865.2 Member Typedef Documentation

4.865.2.1 typedef `_Tp` \* `std::binary_function<_Tp *, _Arg, _Ret>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.865.2.2 `typedef _Ret std::binary_function< _Tp*, _Arg, _Ret >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.865.2.3 `typedef _Arg std::binary_function< _Tp*, _Arg, _Ret >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

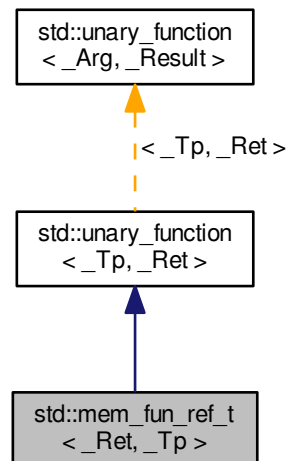
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.866 std::mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference

Inheritance diagram for `std::mem_fun_ref_t< _Ret, _Tp >`:



### Public Types

- `typedef _Tp` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

### Public Member Functions

- `mem_fun_ref_t` (`_Ret` (`_Tp::*` `__pf`)())
- `_Ret operator()` (`_Tp &` `__r`) `const`

## 4.866.1 Detailed Description

```
template<typename _Ret, typename _Tp>class std::mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

Definition at line 578 of file stl\_function.h.

## 4.866.2 Member Typedef Documentation

4.866.2.1 `typedef _Tp std::unary_function< _Tp, _Ret >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl\_function.h.

4.866.2.2 `typedef _Ret std::unary_function< _Tp, _Ret >::result_type` [inherited]

`result_type` is the return type

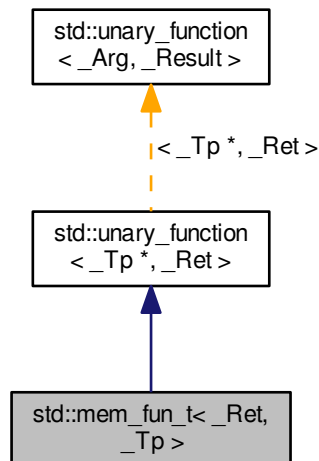
Definition at line 107 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.867 std::mem\_fun\_t&lt; \_Ret, \_Tp &gt; Class Template Reference

Inheritance diagram for `std::mem_fun_t< _Ret, _Tp >`:



## Public Types

- `typedef _Tp * argument\_type`
- `typedef _Ret result\_type`

## Public Member Functions

- `mem_fun_t(_Ret(_Tp::*__pf)())`
- `_Ret operator()(_Tp *__p) const`

### 4.867.1 Detailed Description

`template<typename _Ret, typename _Tp>class std::mem_fun_t<_Ret, _Tp>`

One of the [adaptors for member pointers](#).

Definition at line 542 of file `stl_function.h`.

### 4.867.2 Member Typedef Documentation

4.867.2.1 `typedef _Tp * std::unary_function<_Tp *, _Ret>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.867.2.2 `typedef _Ret std::unary_function<_Tp *, _Ret>::result_type` [inherited]

`result_type` is the return type

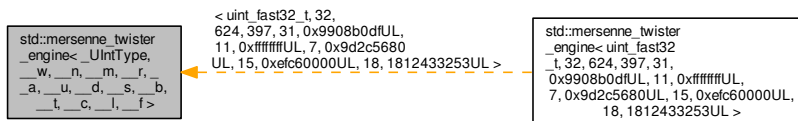
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.868 `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>` **Class Template Reference**

Inheritance diagram for `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>`:



## Public Types

- typedef `_UIntType` `result_type`

## Public Member Functions

- `mersenne_twister_engine` (`result_type` \_\_sd=default\_seed)
- template<typename \_Sseq, typename = typename std::enable\_if<!std::is\_same<\_Sseq, mersenne\_twister\_engine>::value>::type> `mersenne_twister_engine` (\_Sseq &\_\_q)
- void `discard` (unsigned long long \_\_z)
- `result_type` `operator()` ()
- void `seed` (`result_type` \_\_sd=default\_seed)
- template<typename \_Sseq > `std::enable_if< std::is_class <_Sseq>::value>::type` `seed` (\_Sseq &\_\_q)

## Static Public Member Functions

- static constexpr `result_type` `max` ()
- static constexpr `result_type` `min` ()

## Static Public Attributes

- static constexpr `result_type` `default_seed`
- static constexpr `result_type` `initialization_multiplier`
- static constexpr `size_t` `mask_bits`
- static constexpr `size_t` `shift_size`
- static constexpr `size_t` `state_size`
- static constexpr `result_type` `tempering_b`
- static constexpr `result_type` `tempering_c`
- static constexpr `result_type` `tempering_d`
- static constexpr `size_t` `tempering_l`
- static constexpr `size_t` `tempering_s`
- static constexpr `size_t` `tempering_t`
- static constexpr `size_t` `tempering_u`
- static constexpr `size_t` `word_size`
- static constexpr `result_type` `xor_mask`

## Friends

- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_n1, size\_t \_\_m1, size\_t \_\_r1, \_UIntType1 \_\_a1, size\_t \_\_u1, \_UIntType1 \_\_d1, size\_t \_\_s1, \_UIntType1 \_\_b1, size\_t \_\_t1, \_UIntType1 \_\_c1, size\_t \_\_l1, \_UIntType1 \_\_f1, typename \_CharT, typename \_Traits > `std::basic_ostream<_CharT, _Traits>` & `operator<<` (`std::basic_ostream<_CharT, _Traits>` &\_\_os, const `std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1>` &\_\_x)
- bool `operator==` (const `mersenne_twister_engine` &\_\_lhs, const `mersenne_twister_engine` &\_\_rhs)
- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_n1, size\_t \_\_m1, size\_t \_\_r1, \_UIntType1 \_\_a1, size\_t \_\_u1, \_UIntType1 \_\_d1, size\_t \_\_s1, \_UIntType1 \_\_b1, size\_t \_\_t1, \_UIntType1 \_\_c1, size\_t \_\_l1, \_UIntType1 \_\_f1, typename \_CharT, typename \_Traits > `std::basic_istream<_CharT, _Traits>` & `operator>>` (`std::basic_istream<_CharT, _Traits>` &\_\_is, `std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1>` &\_\_x)

#### 4.868.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType
__b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,
__s, __b, __t, __c, __l, __f >
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined mt19937 class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

#### Template Parameters

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>__w</code> | Word size, the number of bits in each element of the state vector. |
| <code>__n</code> | The degree of recursion.                                           |
| <code>__m</code> | The period parameter.                                              |
| <code>__r</code> | The separation point bit index.                                    |
| <code>__a</code> | The last row of the twist matrix.                                  |
| <code>__u</code> | The first right-shift tempering matrix parameter.                  |
| <code>__d</code> | The first right-shift tempering matrix mask.                       |
| <code>__s</code> | The first left-shift tempering matrix parameter.                   |
| <code>__b</code> | The first left-shift tempering matrix mask.                        |
| <code>__t</code> | The second left-shift tempering matrix parameter.                  |
| <code>__c</code> | The second left-shift tempering matrix mask.                       |
| <code>__l</code> | The second right-shift tempering matrix parameter.                 |
| <code>__f</code> | Initialization multiplier.                                         |

Definition at line 449 of file random.h.

#### 4.868.2 Member Typedef Documentation

4.868.2.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type`

The type of the generated random value.

Definition at line 452 of file random.h.

#### 4.868.3 Constructor & Destructor Documentation

4.868.3.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value>::type> std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine ( _Sseq & __q ) [inline], [explicit]`

Constructs a mersenne\_twister\_engine random number generator engine seeded from the seed sequence `__q`.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Definition at line 513 of file random.h.

#### 4.868.4 Member Function Documentation

4.868.4.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> void std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::discard ( unsigned long long __z )`

Discard a sequence of random numbers.

Definition at line 435 of file bits/random.tcc.

4.868.4.2 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::max ( ) [inline], [static]`

Gets the largest possible value in the output range.

Definition at line 534 of file random.h.

4.868.4.3 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::min ( ) [inline], [static]`

Gets the smallest possible value in the output range.

Definition at line 527 of file random.h.

#### 4.868.5 Friends And Related Function Documentation

4.868.5.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> & __x ) [friend]`

Inserts the current state of a % mersenne\_twister\_engine random number generator engine `__x` into the output stream `__os`.

#### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__os</code> | An output stream.                                           |
| <code>__x</code>  | A % mersenne_twister_engine random number generator engine. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.868.5.2 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool operator==( const mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __lhs, const mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __rhs ) [friend]`

Compares two % mersenne\_twister\_engine random number generator objects of the same type for equality.

#### Parameters

|                    |                                                                   |
|--------------------|-------------------------------------------------------------------|
| <code>__lhs</code> | A % mersenne_twister_engine random number generator object.       |
| <code>__rhs</code> | Another % mersenne_twister_engine random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 559 of file random.h.

4.868.5.3 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> & __x ) [friend]`

Extracts the current state of a % mersenne\_twister\_engine random number generator engine `__x` from the input stream `__is`.

#### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__is</code> | An input stream.                                            |
| <code>__x</code>  | A % mersenne_twister_engine random number generator engine. |

#### Returns

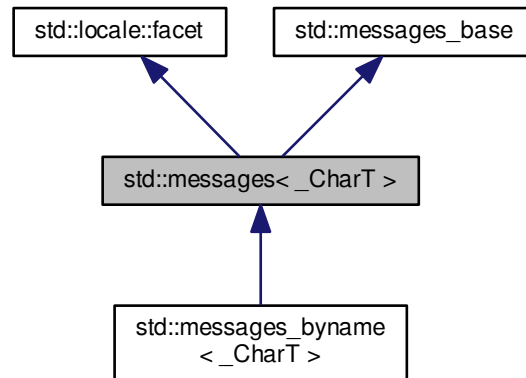
The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.869 std::messages&lt; \_CharT &gt; Class Template Reference

Inheritance diagram for std::messages< \_CharT >:



## Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef **basic\_string**< `_CharT` > **string\_type**

## Public Member Functions

- **messages** (size\_t \_\_refs=0)
- **messages** (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- **string\_type** **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const **string\_type** &\_\_s) const
- catalog **open** (const **basic\_string**< char > &\_\_s, const **locale** &\_\_loc) const
- catalog **open** (const **basic\_string**< char > &, const **locale** &, const char \*) const

## Static Public Attributes

- static **locale::id** id

## Protected Member Functions

- virtual **~messages** ()
- **string\_type** **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const **string\_type** &\_\_msg) const
- virtual void **do\_close** (catalog) const

- virtual [string\\_type](#) **do\_get** (catalog, int, int, const [string\\_type](#) &\_\_default) const
- template<>  
[string](#) **do\_get** (catalog, int, int, const [string](#) &) const
- template<>  
[wstring](#) **do\_get** (catalog, int, int, const [wstring](#) &) const
- virtual catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const

#### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_messages**
- const char \* **\_M\_name\_messages**

#### 4.869.1 Detailed Description

template<typename \_CharT>class std::messages<\_CharT>

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar\_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1695 of file locale\_facets\_nonio.h.

#### 4.869.2 Member Typedef Documentation

4.869.2.1 template<typename \_CharT> typedef \_CharT std::messages<\_CharT>::char\_type

Public typedefs.

Definition at line 1701 of file locale\_facets\_nonio.h.

4.869.2.2 template<typename \_CharT> typedef basic\_string<\_CharT> std::messages<\_CharT>::string\_type

Public typedefs.

Definition at line 1702 of file locale\_facets\_nonio.h.

## 4.869.3 Constructor &amp; Destructor Documentation

4.869.3.1 `template<typename _CharT> std::messages<_CharT>::messages ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 44 of file messages\_members.h.

4.869.3.2 `template<typename _CharT> std::messages<_CharT>::messages ( __c_locale __cloc, const char * __s, size_t __refs = 0 ) [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__s</code>    | The name of a locale.               |
| <code>__refs</code> | Refcount to pass to the base class. |

Definition at line 50 of file messages\_members.h.

4.869.3.3 `template<typename _CharT> std::messages<_CharT>::~~messages ( ) [protected], [virtual]`

Destructor.

Definition at line 79 of file messages\_members.h.

## 4.869.4 Member Data Documentation

4.869.4.1 `template<typename _CharT> locale::id std::messages<_CharT>::id [static]`

Numpunct facet id.

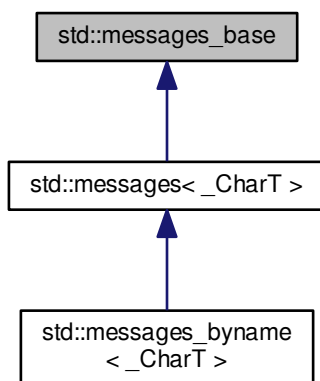
Definition at line 1713 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 4.870 std::messages\_base Struct Reference

Inheritance diagram for std::messages\_base:



### Public Types

- typedef int **catalog**

### 4.870.1 Detailed Description

Messages facet base class providing catalog typedef.

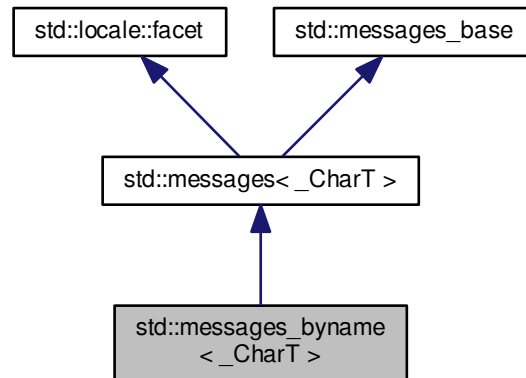
Definition at line 1668 of file `locale_facets_nonio.h`.

The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.871 std::messages\_byname&lt; \_CharT &gt; Class Template Reference

Inheritance diagram for std::messages\_byname< \_CharT >:



## Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string< _CharT >` **string\_type**

## Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- `string_type` **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const `string_type` &\_\_s) const
- catalog **open** (const `basic_string`< char > &\_\_s, const `locale` &\_\_loc) const
- catalog **open** (const `basic_string`< char > &, const `locale` &, const char \*) const

## Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- `string_type` **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const `string_type` &\_\_msg) const
- virtual void **do\_close** (catalog) const
- virtual `string_type` **do\_get** (catalog, int, int, const `string_type` &\_\_dfault) const
- template<>  
`string` **do\_get** (catalog, int, int, const `string` &) const

- `template<>`  
`wstring do_get` (catalog, int, int, const `wstring` &) const
- virtual catalog `do_open` (const `basic_string`< char > &, const `locale` &) const

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale` &\_\_cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` &\_\_cloc, const char \*\_\_s, `__c_locale` \_\_old=0)
- static void `_S_destroy_c_locale` (`__c_locale` &\_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale` \_\_cloc, const char \*\_\_s)

#### Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char \* `_M_name_messages`

#### 4.871.1 Detailed Description

`template<typename _CharT>class std::messages_byname<_CharT>`

class `messages_byname` [22.2.7.2].

Definition at line 1879 of file `locale_facets_nonio.h`.

#### 4.871.2 Member Data Documentation

4.871.2.1 `template<typename _CharT> locale::id std::messages<_CharT>::id` [static], [inherited]

Numpunct facet id.

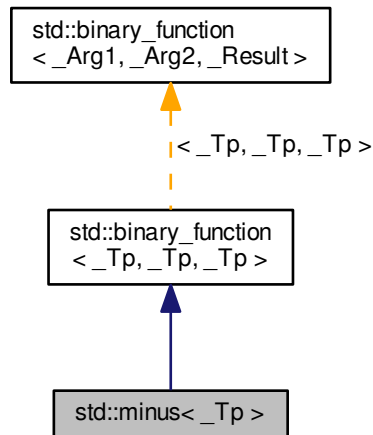
Definition at line 1713 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 4.872 std::minus&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::minus< \_Tp >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.872.1 Detailed Description

```
template<typename _Tp>struct std::minus< _Tp >
```

One of the [math functors](#).

Definition at line 149 of file `stl_function.h`.

## 4.872.2 Member Typedef Documentation

4.872.2.1 typedef `_Tp` `std::binary_function<_Tp, _Tp, _Tp>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.872.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.872.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

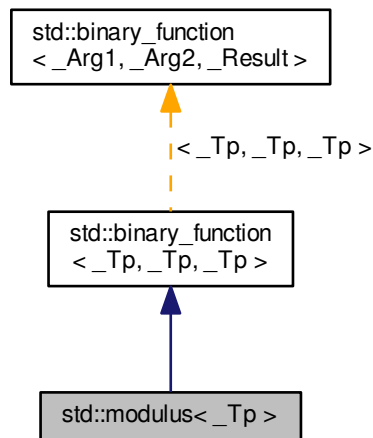
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.873 std::modulus< \_Tp > Struct Template Reference

Inheritance diagram for `std::modulus< _Tp >`:



### Public Types

- `typedef _Tp` [first\\_argument\\_type](#)
- `typedef _Tp` [result\\_type](#)
- `typedef _Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.873.1 Detailed Description

```
template<typename _Tp>struct std::modulus< _Tp >
```

One of the [math functors](#).

Definition at line 176 of file stl\_function.h.

## 4.873.2 Member Typedef Documentation

4.873.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.873.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file stl\_function.h.

4.873.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

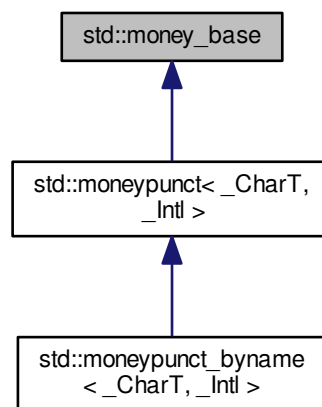
Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.874 std::money\_base Class Reference

Inheritance diagram for `std::money_base`:



## Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- enum **part** {  
    **none**, **space**, **symbol**, **sign**,  
    **value** }

## Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**

## 4.874.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

## See Also

money\_punct::pos\_format() and money\_punct::neg\_format() for details of how these fields are interpreted.

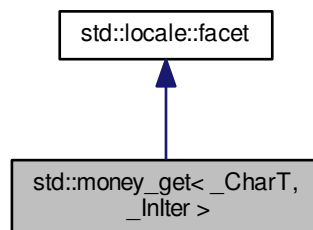
Definition at line 840 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.875 std::money\_get&lt; \_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for std::money\_get< \_CharT, \_InIter >:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`
- typedef `basic_string< _CharT >` `string_type`

## Public Member Functions

- `money_get` (`size_t` \_\_refs=0)
- `iter_type get` (`iter_type` \_\_s, `iter_type` \_\_end, `bool` \_\_intl, `ios_base &` \_\_io, `ios_base::iostate &` \_\_err, `long double &` \_\_units) `const`
- `iter_type get` (`iter_type` \_\_s, `iter_type` \_\_end, `bool` \_\_intl, `ios_base &` \_\_io, `ios_base::iostate &` \_\_err, `string_type &` \_\_digits) `const`

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_get` ()
- `template<bool _Intl>`  
`iter_type _M_extract` (`iter_type` \_\_s, `iter_type` \_\_end, `ios_base &` \_\_io, `ios_base::iostate &` \_\_err, `string &` \_\_digits) `const`
- virtual `iter_type do_get` (`iter_type` \_\_s, `iter_type` \_\_end, `bool` \_\_intl, `ios_base &` \_\_io, `ios_base::iostate &` \_\_err, `long double &` \_\_units) `const`
- virtual `iter_type do_get` (`iter_type` \_\_s, `iter_type` \_\_end, `bool` \_\_intl, `ios_base &` \_\_io, `ios_base::iostate &` \_\_err, `string_type &` \_\_digits) `const`

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (`_c_locale &` \_\_cloc) `throw ()`
- static `void _S_create_c_locale` (`_c_locale &` \_\_cloc, `const char *` \_\_s, `_c_locale` \_\_old=0)
- static `void _S_destroy_c_locale` (`_c_locale &` \_\_cloc)
- static `_c_locale _S_get_c_locale` ()
- static `const char * _S_get_c_name` () `throw ()`
- static `_c_locale _S_lc_ctype_c_locale` (`_c_locale` \_\_cloc, `const char *` \_\_s)

## 4.875.1 Detailed Description

`template<typename _CharT, typename _InIter>class std::money_get< _CharT, _InIter >`

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

The `money_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_get` facet.

Definition at line 1370 of file `locale_facets_nonio.h`.

## 4.875.2 Member Typedef Documentation

## 4.875.2.1 template&lt;typename \_CharT, typename \_Inlter &gt; typedef \_CharT std::money\_get&lt; \_CharT, \_Inlter &gt;::char\_type

Public typedefs.

Definition at line 1376 of file locale\_facets\_nonio.h.

## 4.875.2.2 template&lt;typename \_CharT, typename \_Inlter &gt; typedef \_Inlter std::money\_get&lt; \_CharT, \_Inlter &gt;::iter\_type

Public typedefs.

Definition at line 1377 of file locale\_facets\_nonio.h.

## 4.875.2.3 template&lt;typename \_CharT, typename \_Inlter &gt; typedef basic\_string&lt;\_CharT&gt; std::money\_get&lt; \_CharT, \_Inlter &gt;::string\_type

Public typedefs.

Definition at line 1378 of file locale\_facets\_nonio.h.

## 4.875.3 Constructor &amp; Destructor Documentation

## 4.875.3.1 template&lt;typename \_CharT, typename \_Inlter &gt; std::money\_get&lt; \_CharT, \_Inlter &gt;::money\_get ( size\_t \_\_refs = 0 ) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1392 of file locale\_facets\_nonio.h.

## 4.875.3.2 template&lt;typename \_CharT, typename \_Inlter &gt; virtual std::money\_get&lt; \_CharT, \_Inlter &gt;::~~money\_get ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 1460 of file locale\_facets\_nonio.h.

## 4.875.4 Member Function Documentation

## 4.875.4.1 template&lt;typename \_CharT, typename \_Inlter &gt; \_Inlter std::money\_get&lt; \_CharT, \_Inlter &gt;::do\_get ( iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base &amp; \_\_io, ios\_base::iostate &amp; \_\_err, long double &amp; \_\_units ) const [protected], [virtual]

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

**See Also**

get() for details.

Definition at line 365 of file locale\_facets\_nonio.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str().

Referenced by std::money\_get< \_CharT, \_Inlter >::get().

**4.875.4.2** template<typename \_CharT, typename \_Inlter > \_Inlter std::money\_get< \_CharT, \_Inlter >::do\_get ( iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string\_type & \_\_digits ) const  
[protected], [virtual]

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

**See Also**

get() for details.

Definition at line 378 of file locale\_facets\_nonio.tcc.

References std::ios\_base::M\_getloc(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::resize(), and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

**4.875.4.3** template<typename \_CharT, typename \_Inlter > iter\_type std::money\_get< \_CharT, \_Inlter >::get ( iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long double & \_\_units ) const  
[inline]

Read and parse a monetary value.

This function reads characters from \_\_s, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *units* as an integral value moneypunct::frac\_digits() \* the actual amount. For example, the string \$10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit). *units* is unchanged if parsing fails.

This function works by returning the result of do\_get().

**Parameters**

|         |                                                  |
|---------|--------------------------------------------------|
| __s     | Start of characters to parse.                    |
| __end   | End of characters to parse.                      |
| __intl  | Parameter to use_facet<moneypunct<CharT,intl> >. |
| __io    | Source of facets and io state.                   |
| __err   | Error field to set if parsing fails.             |
| __units | Place to store result of parsing.                |

**Returns**

Iterator referencing first character beyond valid money amount.

Definition at line 1422 of file locale\_facets\_nonio.h.

References std::money\_get< \_CharT, \_Inlter >::do\_get().

```
4.875.4.4 template<typename _CharT, typename _InIter> iter_type std::money_get< _CharT, _InIter >::get (iter_type
 __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, string_type & __digits) const
 [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

#### Parameters

|                       |                                                                            |
|-----------------------|----------------------------------------------------------------------------|
| <code>__s</code>      | Start of characters to parse.                                              |
| <code>__end</code>    | End of characters to parse.                                                |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                             |
| <code>__err</code>    | Error field to set if parsing fails.                                       |
| <code>__digits</code> | Place to store result of parsing.                                          |

#### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1453 of file `locale_facets_nonio.h`.

References `std::money_get< _CharT, _InIter >::do_get()`.

### 4.875.5 Member Data Documentation

```
4.875.5.1 template<typename _CharT, typename _InIter> locale::id std::money_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

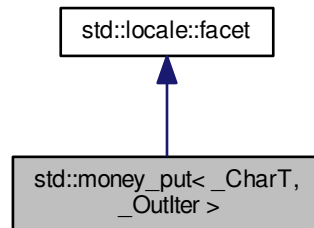
Definition at line 1382 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 4.876 std::money\_put&lt; \_CharT, \_OutIter &gt; Class Template Reference

Inheritance diagram for std::money\_put< \_CharT, \_OutIter >:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`
- typedef `basic_string< _CharT >` `string_type`

## Public Member Functions

- `money_put` (`size_t __refs=0`)
- `iter_type put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `long double __units`) `const`
- `iter_type put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_put` ()
- template<bool \_Intl>  
`iter_type _M_insert` (`iter_type __s`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`
- virtual `iter_type do_put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `long double __units`) `const`
- virtual `iter_type do_put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale & __cloc`) `throw ()`
- static void `_S_create_c_locale` (`__c_locale & __cloc`, `const char * __s`, `__c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale & __cloc`)

- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 4.876.1 Detailed Description

template<typename \_CharT, typename \_Outiter>class std::money\_put< \_CharT, \_Outiter >

Primary class template money\_put.

This facet encapsulates the code to format and output a monetary amount.

The money\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money\_put facet.

Definition at line 1521 of file locale\_facets\_nonio.h.

#### 4.876.2 Member Typedef Documentation

4.876.2.1 template<typename \_CharT, typename \_Outiter> typedef \_CharT std::money\_put< \_CharT, \_Outiter >::char\_type

Public typedefs.

Definition at line 1526 of file locale\_facets\_nonio.h.

4.876.2.2 template<typename \_CharT, typename \_Outiter> typedef \_Outiter std::money\_put< \_CharT, \_Outiter >::iter\_type

Public typedefs.

Definition at line 1527 of file locale\_facets\_nonio.h.

4.876.2.3 template<typename \_CharT, typename \_Outiter> typedef basic\_string<\_CharT> std::money\_put< \_CharT, \_Outiter >::string\_type

Public typedefs.

Definition at line 1528 of file locale\_facets\_nonio.h.

#### 4.876.3 Constructor & Destructor Documentation

4.876.3.1 template<typename \_CharT, typename \_Outiter> std::money\_put< \_CharT, \_Outiter >::money\_put ( size\_t \_\_refs = 0 ) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1542 of file locale\_facets\_nonio.h.

4.876.3.2 `template<typename _CharT, typename _Outlter > virtual std::money_put< _CharT, _Outlter >::~~money_put ( )`  
`[inline], [protected], [virtual]`

Destructor.

Definition at line 1592 of file `locale_facets_nonio.h`.

#### 4.876.4 Member Function Documentation

4.876.4.1 `template<typename _CharT, typename _Outlter > _Outlter std::money_put< _CharT, _Outlter >::do_put ( iter_type`  
`__s, bool __intl, ios_base & __io, char_type __fill, long double __units ) const` `[protected], [virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write \$10.01 to `__s`.

This function is a hook for derived classes to change the value returned.

#### See Also

`put()`.

#### Parameters

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>     | The stream to write to.                                                     |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                              |
| <code>__fill</code>  | <code>char_type</code> to use for padding.                                  |
| <code>__units</code> | Place to store result of parsing.                                           |

#### Returns

Iterator after writing.

Definition at line 570 of file `locale_facets_nonio.tcc`.

References `std::ios_base::getloc()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::money_put< _CharT, _Outlter >::put()`.

4.876.4.2 `template<typename _CharT, typename _Outlter > _Outlter std::money_put< _CharT, _Outlter >::do_put ( iter_type`  
`__s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits ) const` `[protected],`  
`[virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string 1001 in a US locale would write \$10.01 to `__s`.

This function is a hook for derived classes to change the value returned.

#### See Also

`put()`.

## Parameters

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>      | The stream to write to.                                                     |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                              |
| <code>__fill</code>   | <code>char_type</code> to use for padding.                                  |
| <code>__digits</code> | Place to store result of parsing.                                           |

## Returns

Iterator after writing.

Definition at line 608 of file `locale_facets_nonio.tcc`.

4.876.4.3 `template<typename _CharT, typename _Outlter > iter_type std::money_put<_CharT, _Outlter >::put ( iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units ) const [inline]`

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write \$10.01 to `__s`.

This function works by returning the result of `do_put()`.

## Parameters

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>     | The stream to write to.                                                     |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                              |
| <code>__fill</code>  | <code>char_type</code> to use for padding.                                  |
| <code>__units</code> | Place to store result of parsing.                                           |

## Returns

Iterator after writing.

Definition at line 1562 of file `locale_facets_nonio.h`.

References `std::money_put<_CharT, _Outlter >::do_put()`.

4.876.4.4 `template<typename _CharT, typename _Outlter > iter_type std::money_put<_CharT, _Outlter >::put ( iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits ) const [inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string 1001 in a US locale would write \$10.01 to `__s`.

This function works by returning the result of `do_put()`.

## Parameters

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>      | The stream to write to.                                                     |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                              |
| <code>__fill</code>   | <code>char_type</code> to use for padding.                                  |
| <code>__digits</code> | Place to store result of parsing.                                           |

**Returns**

Iterator after writing.

Definition at line 1585 of file locale\_facets\_nonio.h.

References std::money\_put< \_CharT, \_Outiter >::do\_put().

**4.876.5 Member Data Documentation**

4.876.5.1 template<typename \_CharT, typename \_Outiter > locale::id std::money\_put< \_CharT, \_Outiter >::id [static]

Numpunct facet id.

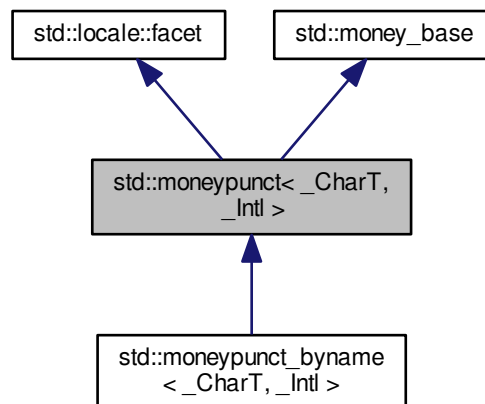
Definition at line 1532 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

**4.877 std::moneypunct< \_CharT, \_Intl > Class Template Reference**

Inheritance diagram for std::moneypunct< \_CharT, \_Intl >:

**Public Types**

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache< \_CharT, \_Intl > **\_\_cache\_type**
- enum **part** { **none**, **space**, **symbol**, **sign**, **value** }

- typedef [\\_CharT](#) [char\\_type](#)
- typedef [basic\\_string](#)< [\\_CharT](#) > [string\\_type](#)

#### Public Member Functions

- [moneypunct](#) (size\_t \_\_refs=0)
- [moneypunct](#) (\_\_cache\_type \* \_\_cache, size\_t \_\_refs=0)
- [moneypunct](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s, size\_t \_\_refs=0)
- [string\\_type](#) [curr\\_symbol](#) () const
- [char\\_type](#) [decimal\\_point](#) () const
- int [frac\\_digits](#) () const
- [string](#) [grouping](#) () const
- [string\\_type](#) [negative\\_sign](#) () const
- [string\\_type](#) [positive\\_sign](#) () const
- [char\\_type](#) [thousands\\_sep](#) () const
- pattern [pos\\_format](#) () const
- pattern [neg\\_format](#) () const

#### Static Public Member Functions

- static pattern [\\_S\\_construct\\_pattern](#) (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

#### Static Public Attributes

- static const char \* [\\_S\\_atoms](#)
- static const pattern [\\_S\\_default\\_pattern](#)
- static [locale::id](#) [id](#)
- static const bool [intl](#)

#### Protected Member Functions

- virtual [~moneypunct](#) ()
- void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale \_\_cloc=0, const char \* \_\_name=0)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- virtual [string\\_type](#) [do\\_curr\\_symbol](#) () const
- virtual [char\\_type](#) [do\\_decimal\\_point](#) () const
- virtual int [do\\_frac\\_digits](#) () const
- virtual [string](#) [do\\_grouping](#) () const
- virtual pattern [do\\_neg\\_format](#) () const
- virtual [string\\_type](#) [do\\_negative\\_sign](#) () const
- virtual pattern [do\\_pos\\_format](#) () const
- virtual [string\\_type](#) [do\\_positive\\_sign](#) () const
- virtual [char\\_type](#) [do\\_thousands\\_sep](#) () const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.877.1 Detailed Description

```
template<typename _CharT, bool _Intl>class std::moneypunct< _CharT, _Intl >
```

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 934 of file locale\_facets\_nonio.h.

## 4.877.2 Member Typedef Documentation

4.877.2.1 `template<typename _CharT, bool _Intl> typedef _CharT std::moneypunct< _CharT, _Intl >::char_type`

Public typedefs.

Definition at line 940 of file locale\_facets\_nonio.h.

4.877.2.2 `template<typename _CharT, bool _Intl> typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type`

Public typedefs.

Definition at line 941 of file locale\_facets\_nonio.h.

## 4.877.3 Constructor &amp; Destructor Documentation

4.877.3.1 `template<typename _CharT, bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct ( size_t __refs = 0 ) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 963 of file locale\_facets\_nonio.h.

4.877.3.2 `template<typename _CharT, bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct ( __cache_type * __cache, size_t __refs = 0 ) [inline], [explicit]`

Constructor performs initialization.

This is an internal constructor.

## Parameters

|                      |                                 |
|----------------------|---------------------------------|
| <code>__cache</code> | Cache for optimization.         |
| <code>__refs</code>  | Passed to the base facet class. |

Definition at line 976 of file locale\_facets\_nonio.h.

**4.877.3.3** `template<typename _CharT, bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct ( __c.locale __cloc, const char * __s, size_t __refs = 0 ) [inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__s</code>    | The name of a locale.           |
| <code>__refs</code> | Passed to the base facet class. |

Definition at line 991 of file locale\_facets\_nonio.h.

**4.877.3.4** `template<typename _CharT, bool _Intl> virtual std::moneypunct< _CharT, _Intl >::~~moneypunct ( ) [protected], [virtual]`

Destructor.

**4.877.4 Member Function Documentation**

**4.877.4.1** `template<typename _CharT, bool _Intl> string_type std::moneypunct< _CharT, _Intl >::curr_symbol ( ) const [inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `moneypunct<char_type>::do_curr_symbol()`.

## Returns

*string\_type* representing a currency symbol.

Definition at line 1061 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

**4.877.4.2** `template<typename _CharT, bool _Intl> char_type std::moneypunct< _CharT, _Intl >::decimal_point ( ) const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `moneypunct<char_type>::do_decimal_point()`.

## Returns

*char\_type* representing a decimal point.

Definition at line 1005 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

**4.877.4.3** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol ( )  
const [inline], [protected], [virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

#### See Also

`curr_symbol()` for details.

#### Returns

*string\_type* representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::curr_symbol()`.

**4.877.4.4** `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point ( )  
const [inline], [protected], [virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::decimal_point()`.

**4.877.4.5** `template<typename _CharT, bool _Intl> virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits ( ) const  
[inline], [protected], [virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

#### See Also

`frac_digits()` for details.

#### Returns

Number of digits in amount fraction.

Definition at line 1247 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::frac_digits()`.

**4.877.4.6** `template<typename _CharT, bool _Intl> virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const  
[inline], [protected], [virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See Also**

grouping() for details.

**Returns**

String representing grouping specification.

Definition at line 1194 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

**4.877.4.7** `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const`  
`[inline], [protected], [virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See Also**

neg\_format() for details.

**Returns**

Pattern for money values.

Definition at line 1275 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::neg\_format().

**4.877.4.8** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const`  
`[inline], [protected], [virtual]`

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See Also**

negative\_sign() for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1233 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::negative\_sign().

**4.877.4.9** `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const`  
`[inline], [protected], [virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See Also**

pos\_format() for details.

**Returns**

Pattern for money values.

Definition at line 1261 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::pos\_format().

**4.877.4.10** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const [inline], [protected], [virtual]`

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See Also**

positive\_sign() for details.

**Returns**

string\_type representing a positive sign.

Definition at line 1220 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::positive\_sign().

**4.877.4.11** `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep ( ) const [inline], [protected], [virtual]`

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

char\_type representing a thousands separator.

Definition at line 1181 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::thousands\_sep().

**4.877.4.12** `template<typename _CharT, bool _Intl> int std::moneypunct< _CharT, _Intl >::frac_digits ( ) const [inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char\_type>::do\_frac\_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac\_digits() digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1111 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_frac\_digits().

**4.877.4.13** `template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl>::grouping ( ) const`  
`[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

#### Returns

string representing grouping specification.

Definition at line 1048 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_grouping()`.

**4.877.4.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format ( ) const`  
`[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_neg_format()`.

**4.877.4.15** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::negative_sign ( ) const`  
`[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_negative_sign()`.

**4.877.4.16** `template<typename _CharT, bool _Intl> pattern std::moneypunct< _CharT, _Intl >::pos_format ( ) const`  
`[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_pos_format()`.

**4.877.4.17** `template<typename _CharT, bool _Intl> string_type std::moneypunct< _CharT, _Intl >::positive_sign ( ) const`  
`[inline]`

Return positive sign string.

This function returns a *string\_type* to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_positive_sign()`.

**4.877.4.18** `template<typename _CharT, bool _Intl> char_type std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const`  
`[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

#### Returns

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_thousands_sep()`.

#### 4.877.5 Member Data Documentation

4.877.5.1 `template<typename _CharT, bool _Intl> locale::id std::moneypunct< _CharT, _Intl >::id [static]`

Numpunct facet id.

Definition at line 953 of file `locale_facets_nonio.h`.

4.877.5.2 `template<typename _CharT, bool _Intl> const bool std::moneypunct< _CharT, _Intl >::intl [static]`

This value is provided by the standard, but no reason for its existence.

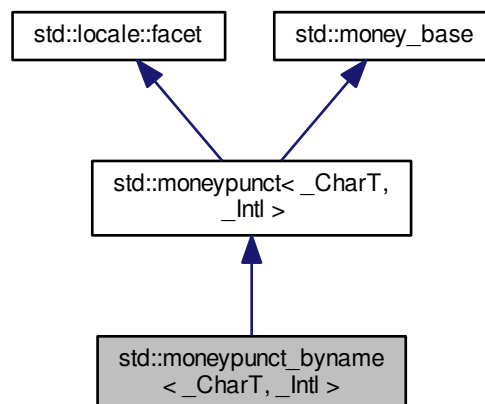
Definition at line 951 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.878 std::moneypunct\_byname< \_CharT, \_Intl > Class Template Reference

Inheritance diagram for `std::moneypunct_byname< _CharT, _Intl >`:



## Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache<\_CharT, \_Intl> **\_\_cache\_type**
- typedef \_CharT **char\_type**
- enum **part** {  
    **none**, **space**, **symbol**, **sign**,  
    **value** }
- typedef basic\_string<\_CharT> **string\_type**

## Public Member Functions

- **moneypunct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- [string\\_type curr\\_symbol](#) () const
- [char\\_type decimal\\_point](#) () const
- int [frac\\_digits](#) () const
- [string grouping](#) () const
- [string\\_type negative\\_sign](#) () const
- [string\\_type positive\\_sign](#) () const
- [char\\_type thousands\\_sep](#) () const
- pattern [pos\\_format](#) () const
- pattern [neg\\_format](#) () const

## Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

## Protected Member Functions

- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<>  
    void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
    void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
    void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
    void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type do\\_curr\\_symbol](#) () const
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual int [do\\_frac\\_digits](#) () const

- virtual [string do\\_grouping](#) () const
- virtual pattern [do\\_neg\\_format](#) () const
- virtual [string\\_type do\\_negative\\_sign](#) () const
- virtual pattern [do\\_pos\\_format](#) () const
- virtual [string\\_type do\\_positive\\_sign](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const

#### Static Protected Member Functions

- static `__c_locale` [\\_S\\_clone\\_c\\_locale](#) (`__c_locale &__cloc`) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (`__c_locale &__cloc`, const char \*`__s`, `__c_locale __old=0`)
- static void [\\_S\\_destroy\\_c\\_locale](#) (`__c_locale &__cloc`)
- static `__c_locale` [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static `__c_locale` [\\_S\\_lc\\_type\\_c\\_locale](#) (`__c_locale __cloc`, const char \*`__s`)

#### 4.878.1 Detailed Description

template<typename `_CharT`, bool `_Intl`>class std::moneypunct\_byname< `_CharT`, `_Intl` >

class moneypunct\_byname [22.2.6.4].

Definition at line 1324 of file locale\_facets\_nonio.h.

#### 4.878.2 Member Function Documentation

4.878.2.1 template<typename `_CharT`, bool `_Intl`> [string\\_type std::moneypunct< `\_CharT`, `\_Intl` >::curr\\_symbol \( \) const](#)  
[[inline](#)], [[inherited](#)]

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `moneypunct<char_type>::do_curr_symbol()`.

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1061 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

4.878.2.2 template<typename `_CharT`, bool `_Intl`> [char\\_type std::moneypunct< `\_CharT`, `\_Intl` >::decimal\\_point \( \) const](#)  
[[inline](#)], [[inherited](#)]

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `moneypunct<char_type>::do_decimal_point()`.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1005 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

**4.878.2.3** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_curr_symbol ( ) const [inline], [protected], [virtual], [inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

#### See Also

`curr_symbol()` for details.

#### Returns

*string\_type* representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::curr_symbol()`.

**4.878.2.4** `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_decimal_point ( ) const [inline], [protected], [virtual], [inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::decimal_point()`.

**4.878.2.5** `template<typename _CharT, bool _Intl> virtual int std::moneypunct<_CharT, _Intl>::do_frac_digits ( ) const [inline], [protected], [virtual], [inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

#### See Also

`frac_digits()` for details.

#### Returns

Number of digits in amount fraction.

Definition at line 1247 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::frac_digits()`.

**4.878.2.6** `template<typename _CharT, bool _Intl> virtual string std::moneypunct<_CharT, _Intl>::do_grouping ( ) const [inline], [protected], [virtual], [inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See Also**

grouping() for details.

**Returns**

String representing grouping specification.

Definition at line 1194 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

**4.878.2.7** `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See Also**

neg\_format() for details.

**Returns**

Pattern for money values.

Definition at line 1275 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::neg\_format().

**4.878.2.8** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See Also**

negative\_sign() for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1233 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::negative\_sign().

**4.878.2.9** `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const`  
`[inline], [protected], [virtual], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See Also**

pos\_format() for details.

**Returns**

Pattern for money values.

Definition at line 1261 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::pos\_format().

**4.878.2.10** `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_positive_sign ( ) const [inline], [protected], [virtual], [inherited]`

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See Also**

positive\_sign() for details.

**Returns**

string\_type representing a positive sign.

Definition at line 1220 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::positive\_sign().

**4.878.2.11** `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep ( ) const [inline], [protected], [virtual], [inherited]`

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

char\_type representing a thousands separator.

Definition at line 1181 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::thousands\_sep().

**4.878.2.12** `template<typename _CharT, bool _Intl> int std::moneypunct<_CharT, _Intl>::frac_digits ( ) const [inline], [inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char\_type>::do\_frac\_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac\_digits() digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1111 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl >::do\_frac\_digits().

**4.878.2.13** `template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl >::grouping ( ) const`  
`[inline], [inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

Definition at line 1048 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl >::do\_grouping().

**4.878.2.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl >::neg_format ( ) const`  
`[inline], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

**Returns**

Pattern for money values.

Definition at line 1151 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl >::do\_neg\_format().

**4.878.2.15** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::negative_sign ( ) const`  
`[inline], [inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_negative_sign()`.

**4.878.2.16** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::pos_format ( ) const`  
`[inline], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_pos_format()`.

**4.878.2.17** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::positive_sign ( ) const`  
`[inline], [inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

## Returns

*string\_type* representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_positive_sign()`.

4.878.2.18 `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::thousands_sep( ) const`  
`[inline], [inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

## Returns

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_thousands_sep()`.

## 4.878.3 Member Data Documentation

4.878.3.1 `template<typename _CharT, bool _Intl> locale::id std::moneypunct<_CharT, _Intl>::id` `[static], [inherited]`

Numpunct facet id.

Definition at line 953 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

4.879 `std::move_iterator<_Iterator>` Class Template Reference

## Public Types

- typedef `__traits_type::difference_type` **difference\_type**
- typedef `__traits_type::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `_Iterator` **pointer**
- typedef `value_type &&` **reference**
- typedef `__traits_type::value_type` **value\_type**

## Public Member Functions

- **move\_iterator** (`iterator_type __i`)
- `template<typename _Iter>`  
**move\_iterator** (`const move\_iterator<_Iter> &__i`)

- iterator\_type **base** () const
- reference **operator\*** () const
- [move\\_iterator](#) **operator+** (difference\_type \_\_n) const
- [move\\_iterator](#) & **operator++** ()
- [move\\_iterator](#) **operator++** (int)
- [move\\_iterator](#) & **operator+=** (difference\_type \_\_n)
- [move\\_iterator](#) **operator-** (difference\_type \_\_n) const
- [move\\_iterator](#) & **operator--** ()
- [move\\_iterator](#) **operator--** (int)
- [move\\_iterator](#) & **operator-=** (difference\_type \_\_n)
- pointer **operator->** () const
- reference **operator[]** (difference\_type \_\_n) const

#### Protected Types

- typedef iterator\_traits  
< \_Iterator > **\_\_traits\_type**

#### Protected Attributes

- \_Iterator **\_M\_current**

#### 4.879.1 Detailed Description

template<typename \_Iterator>class std::move\_iterator< \_Iterator >

Class template move\_iterator is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 930 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.880 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

#### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef  
\_Pair\_alloc\_type::const\_pointer **const\_pointer**
- typedef  
\_Pair\_alloc\_type::const\_reference **const\_reference**
- typedef  
[\\_Rep\\_type::const\\_reverse\\_iterator](#) **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**

- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Pair_alloc_type::pointer` **pointer**
- typedef `_Pair_alloc_type::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

#### Public Member Functions

- `multimap` ()
- `multimap` (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `multimap` (const `multimap` &\_\_x)
- `multimap` (`multimap` &&\_\_x) noexcept(`is_nothrow_copy_constructible< _Compare >`)
- `multimap` (`initializer_list< value_type >` \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- template<typename `_InputIterator` >  
`multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` >  
`multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- iterator `begin` () noexcept
- const\_iterator `begin` () const noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- size\_type `count` (const `key_type` &\_\_x) const
- const\_reverse\_iterator `crbegin` () const noexcept
- const\_reverse\_iterator `crend` () const noexcept
- template<typename... `_Args` >  
iterator `emplace` (`_Args` &&...\_\_args)
- template<typename... `_Args` >  
iterator `emplace_hint` (const\_iterator \_\_pos, `_Args` &&...\_\_args)
- bool `empty` () const noexcept
- iterator `end` () noexcept
- const\_iterator `end` () const noexcept
- `std::pair< iterator, iterator >` `equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator >` `equal_range` (const `key_type` &\_\_x) const
- iterator `erase` (const\_iterator \_\_position)
- iterator `erase` (iterator \_\_position)
- size\_type `erase` (const `key_type` &\_\_x)
- iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator `find` (const `key_type` &\_\_x)
- const\_iterator `find` (const `key_type` &\_\_x) const
- `allocator_type` `get_allocator` () const noexcept
- iterator `insert` (const `value_type` &\_\_x)
- template<typename `_Pair` , typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator `insert` (`_Pair` &&\_\_x)
- iterator `insert` (const\_iterator \_\_position, const `value_type` &\_\_x)

- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value::type>  
iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- key\_compare **key\_comp** () const
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- size\_type **max\_size** () const noexcept
- multimap & **operator=** (const multimap &\_\_x)
- multimap & **operator=** (multimap &&\_\_x)
- multimap & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- size\_type **size** () const noexcept
- void **swap** (multimap &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- value\_compare **value\_comp** () const

#### Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**< (const multimap< \_K1, \_T1, \_C1, \_A1 > &, const multimap< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator==** (const multimap< \_K1, \_T1, \_C1, \_A1 > &, const multimap< \_K1, \_T1, \_C1, \_A1 > &)

#### 4.880.1 Detailed Description

template<typename \_Key, typename \_Tp, typename \_Compare = std::less<\_Key>, typename \_Alloc = std::allocator<std::pair<const \_Key, \_Tp> >>> class std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

#### Template Parameters

|                       |                                                               |
|-----------------------|---------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                          |
| <code>_Tp</code>      | Type of mapped objects.                                       |
| <code>_Compare</code> | Comparison function object type, defaults to less<_Key>.      |
| <code>_Alloc</code>   | Allocator type, defaults to allocator<pair<const _Key, _Tp>>. |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 95 of file `stl_multimap.h`.

## 4.880.2 Constructor &amp; Destructor Documentation

4.880.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( )`  
`[inline]`

Default constructor creates no elements.

Definition at line 157 of file `stl_multimap.h`.

4.880.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const _Compare & __comp, const allocator_type & __a = allocator_type() )` `[inline],[explicit]`

Creates a multimap with no elements.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

Definition at line 166 of file `stl_multimap.h`.

4.880.2.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const multimap<_Key, _Tp, _Compare, _Alloc> & __x )` `[inline]`

Multimap copy constructor.

## Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multimap of identical element and allocator types. |
|------------------|------------------------------------------------------|

The newly-created multimap uses a copy of the allocation object used by `__x`.

Definition at line 177 of file `stl_multimap.h`.

4.880.2.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( multimap<_Key, _Tp, _Compare, _Alloc> && __x )` `[inline],[noexcept]`

Multimap move constructor.

## Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multimap of identical element and allocator types. |
|------------------|------------------------------------------------------|

The newly-created multimap contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified multimap.

Definition at line 188 of file `stl_multimap.h`.

4.880.2.5 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( initializer_list<value_type> & __l, const _Compare & __comp = _Compare(), const allocator_type & __a = allocator_type() )` `[inline]`

Builds a multimap from an `initializer_list`.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a multimap consisting of copies of the elements from the initializer\_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 202 of file `stl_multimap.h`.

```
4.880.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > std::multimap< _Key, _Tp,
_Compare, _Alloc >::multimap (_InputIterator __first, _InputIterator __last) [inline]
```

Builds a multimap from a range.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multimap consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first,__last)`).

Definition at line 219 of file `stl_multimap.h`.

```
4.880.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > std::multimap< _Key, _Tp,
_Compare, _Alloc >::multimap (_InputIterator __first, _InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from a range.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multimap consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first,__last)`).

Definition at line 235 of file `stl_multimap.h`.

## 4.880.3 Member Function Documentation

```
4.880.3.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ()
[inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 314 of file `stl_multimap.h`.

```
4.880.3.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin (
) const [inline],[noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 323 of file stl\_multimap.h.

```
4.880.3.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cbegin (
) const [inline],[noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 387 of file stl\_multimap.h.

```
4.880.3.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cend (
) const [inline],[noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 396 of file stl\_multimap.h.

```
4.880.3.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::clear ()
[inline],[noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 696 of file stl\_multimap.h.

Referenced by std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::operator=().

```
4.880.3.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::count (
const key_type & __x) const [inline]
```

Finds the number of elements with given key.

#### Parameters

|     |                                          |
|-----|------------------------------------------|
| __x | Key of (key, value) pairs to be located. |
|-----|------------------------------------------|

**Returns**

Number of elements with specified key.

Definition at line 753 of file stl\_multimap.h.

**4.880.3.7** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::crbegin( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 405 of file stl\_multimap.h.

**4.880.3.8** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::crend( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 414 of file stl\_multimap.h.

**4.880.3.9** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::emplace( _Args &&... __args ) [inline]`

Build and insert a std::pair into the multimap.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 454 of file stl\_multimap.h.

**4.880.3.10** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::emplace_hint( const_iterator __pos, _Args &&... __args ) [inline]`

Builds and inserts a std::pair into the multimap.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 481 of file stl\_multimap.h.

```
4.880.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::multimap<_Key, _Tp, _Compare, _Alloc>::empty ()
const [inline], [noexcept]
```

Returns true if the multimap is empty.

Definition at line 421 of file stl\_multimap.h.

```
4.880.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end ()
[inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 332 of file stl\_multimap.h.

```
4.880.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end (
) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 341 of file stl\_multimap.h.

```
4.880.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::multimap<_Key, _Tp, _Compare,
_Alloc>::equal_range (const key_type & _x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 820 of file stl\_multimap.h.

```
4.880.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator> std::multimap< _Key,
_Tp, _Compare, _Alloc >::equal_range (const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

#### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 837 of file stl\_multimap.h.

```
4.880.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __position) [inline]
```

Erases an element from a multimap.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 595 of file stl\_multimap.h.

```
4.880.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
const key_type & __x) [inline]
```

Erases elements according to the provided key.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 630 of file stl\_multimap.h.

```
4.880.3.18 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::erase (
const_iterator __first, const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multimap.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased .  |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 651 of file stl\_multimap.h.

```
4.880.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::find (const
key_type & __x) [inline]
```

Tries to locate an element in a multimap.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 729 of file stl\_multimap.h.

```
4.880.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::find (
const key_type & __x) const [inline]
```

Tries to locate an element in a multimap.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

## Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 744 of file `stl_multimap.h`.

```
4.880.3.21 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc
>::get_allocator() const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 304 of file `stl_multimap.h`.

```
4.880.3.22 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
const value_type & __x) [inline]
```

Inserts a `std::pair` into the multimap.

## Parameters

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

## Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 501 of file `stl_multimap.h`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`.

```
4.880.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
const_iterator __position, const value_type & __x) [inline]
```

Inserts a `std::pair` into the multimap.

## Parameters

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

## Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 535 of file stl\_multimap.h.

```
4.880.3.24 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void std::multimap< _Key, _Tp,
_Compare, _Alloc >::insert (_InputIterator __first, _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 562 of file stl\_multimap.h.

```
4.880.3.25 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of std::pairs into the multimap.

#### Parameters

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <code>__l</code> | A std::initializer_list<value_type> of pairs to be inserted. |
|------------------|--------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 574 of file stl\_multimap.h.

References std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

Referenced by std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

```
4.880.3.26 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> key_compare std::multimap< _Key, _Tp, _Compare, _Alloc
>::key_comp () const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

Definition at line 705 of file stl\_multimap.h.

```
4.880.3.27 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound
(const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 768 of file `stl_multimap.h`.

```
4.880.3.28 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::lower_bound (const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 783 of file `stl_multimap.h`.

```
4.880.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size (
) const [inline], [noexcept]
```

Returns the maximum size of the multimap.

Definition at line 431 of file `stl_multimap.h`.

```
4.880.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare, _Alloc
>::operator= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x) [inline]
```

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multimap of identical element and allocator types. |
|------------------|------------------------------------------------------|

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 258 of file `stl_multimap.h`.

```
4.880.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare, _Alloc
>::operator= (multimap< _Key, _Tp, _Compare, _Alloc > && __x) [inline]
```

Multimap move assignment operator.

#### Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multimap of identical element and allocator types. |
|------------------|------------------------------------------------------|

The contents of `__x` are moved into this multimap (without copying). `__x` is a valid, but unspecified multimap.

Definition at line 273 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

```
4.880.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare, _Alloc
>::operator= (initializer_list< value_type > __l) [inline]
```

Multimap list assignment operator.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

This function fills a multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 294 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

```
4.880.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 350 of file `stl_multimap.h`.

```
4.880.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 359 of file `stl_multimap.h`.

```
4.880.3.35 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 368 of file stl\_multimap.h.

```
4.880.3.36 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 377 of file stl\_multimap.h.

```
4.880.3.37 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size ()
const [inline], [noexcept]
```

Returns the size of the multimap.

Definition at line 426 of file stl\_multimap.h.

```
4.880.3.38 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (
multimap< _Key, _Tp, _Compare, _Alloc > & __x) [inline]
```

Swaps data with another multimap.

#### Parameters

|     |                                                     |
|-----|-----------------------------------------------------|
| __x | A multimap of the same element and allocator types. |
|-----|-----------------------------------------------------|

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 686 of file stl\_multimap.h.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::swap()`.

```
4.880.3.39 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound
(const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|     |                                         |
|-----|-----------------------------------------|
| __x | Key of (key, value) pair to be located. |
|-----|-----------------------------------------|

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 793 of file stl\_multimap.h.

```
4.880.3.40 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::upper_bound (const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|     |                                         |
|-----|-----------------------------------------|
| __x | Key of (key, value) pair to be located. |
|-----|-----------------------------------------|

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 803 of file stl\_multimap.h.

```
4.880.3.41 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> value_compare std::multimap< _Key, _Tp, _Compare, _Alloc
>::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

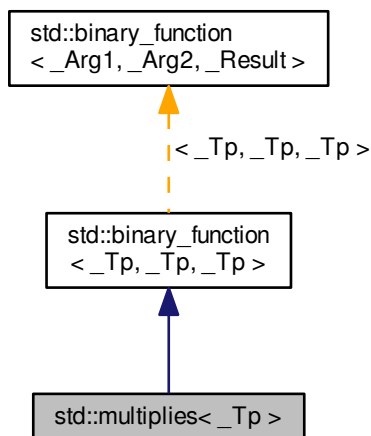
Definition at line 713 of file stl\_multimap.h.

The documentation for this class was generated from the following file:

- [stl\\_multimap.h](#)

## 4.881 std::multiplies< \_Tp > Struct Template Reference

Inheritance diagram for std::multiplies< \_Tp >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.881.1 Detailed Description

`template<typename _Tp>struct std::multiplies< _Tp >`

One of the [math functors](#).

Definition at line 158 of file `stl_function.h`.

## 4.881.2 Member Typedef Documentation

4.881.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.881.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.881.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.882 `std::multiset< _Key, _Compare, _Alloc >` Class Template Reference

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Key_alloc_type::const_pointer` **const\_pointer**
- typedef `_Key_alloc_type::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**

- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::const\_iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Key\_alloc\_type::pointer **pointer**
- typedef \_Key\_alloc\_type::reference **reference**
- typedef  
\_Rep\_type::const\_reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

### Public Member Functions

- **multiset** ()
- **multiset** (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator >  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- **multiset** (const **multiset** &\_\_x)
- **multiset** (**multiset** &&\_\_x) noexcept(**is\_nothrow\_copy\_constructible**< \_Compare >)
- **multiset** (**initializer\_list**< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** () const noexcept
- iterator **cbegin** () const noexcept
- iterator **cend** () const noexcept
- void **clear** () noexcept
- size\_type **count** (const key\_type &\_\_x) const
- **reverse\_iterator** **crbegin** () const noexcept
- **reverse\_iterator** **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&...\_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&...\_\_args)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- iterator **erase** (const\_iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- allocator\_type **get\_allocator** () const noexcept
- iterator **insert** (const value\_type &\_\_x)
- iterator **insert** (value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (**initializer\_list**< value\_type > \_\_l)
- key\_compare **key\_comp** () const
- size\_type **max\_size** () const noexcept

- `multiset` & `operator=` (const `multiset` &\_\_x)
- `multiset` & `operator=` (`multiset` &&\_\_x)
- `multiset` & `operator=` (`initializer_list`< value\_type > \_\_l)
- `reverse_iterator` `rbegin` () const noexcept
- `reverse_iterator` `rend` () const noexcept
- size\_type `size` () const noexcept
- void `swap` (`multiset` &\_\_x)
- value\_compare `value_comp` () const
- iterator `find` (const key\_type &\_\_x)
- const\_iterator `find` (const key\_type &\_\_x) const
- iterator `lower_bound` (const key\_type &\_\_x)
- const\_iterator `lower_bound` (const key\_type &\_\_x) const
- iterator `upper_bound` (const key\_type &\_\_x)
- const\_iterator `upper_bound` (const key\_type &\_\_x) const
- `std::pair`< iterator, iterator > `equal_range` (const key\_type &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const

#### Friends

- template<typename \_K1 , typename \_C1 , typename \_A1 >  
bool **operator**< (const `multiset`< \_K1, \_C1, \_A1 > &, const `multiset`< \_K1, \_C1, \_A1 > &)
- template<typename \_K1 , typename \_C1 , typename \_A1 >  
bool **operator**== (const `multiset`< \_K1, \_C1, \_A1 > &, const `multiset`< \_K1, \_C1, \_A1 > &)

#### 4.882.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Alloc = std::allocator<\_Key>>class std::multiset< \_Key, \_Compare, \_Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time.

#### Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 92 of file `stl_multiset.h`.

## 4.882.2 Constructor &amp; Destructor Documentation

4.882.2.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::multiset< _Key, _Compare, _Alloc >::multiset ( ) [inline]`

Default constructor creates no elements.

Definition at line 137 of file `stl_multiset.h`.

4.882.2.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::multiset< _Key, _Compare, _Alloc >::multiset ( const _Compare & __comp, const allocator_type & __a =  
allocator_type() ) [inline],[explicit]`

Creates a multiset with no elements.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

Definition at line 146 of file `stl_multiset.h`.

4.882.2.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset ( _InputIterator __first,  
_InputIterator __last ) [inline]`

Builds a multiset from a range.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multiset consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first,__last`)).

Definition at line 160 of file `stl_multiset.h`.

4.882.2.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset ( _InputIterator __first,  
_InputIterator __last, const _Compare & __comp, const allocator_type & __a = allocator_type() ) [inline]`

Builds a multiset from a range.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multiset consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first,__last`)).

Definition at line 176 of file `stl_multiset.h`.

```
4.882.2.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (const multiset< _Key, _Compare, _Alloc > & __x)
[inline]
```

Multiset copy constructor.

#### Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multiset of identical element and allocator types. |
|------------------|------------------------------------------------------|

The newly-created multiset uses a copy of the allocation object used by `__x`.

Definition at line 189 of file `stl_multiset.h`.

```
4.882.2.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (multiset< _Key, _Compare, _Alloc > && __x) [inline],
[noexcept]
```

Multiset move constructor.

#### Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multiset of identical element and allocator types. |
|------------------|------------------------------------------------------|

The newly-created multiset contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified multiset.

Definition at line 200 of file `stl_multiset.h`.

```
4.882.2.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (initializer_list< value_type > __l, const _Compare & __comp =
_Compare(), const allocator_type & __a = allocator_type()) [inline]
```

Builds a multiset from an `initializer_list`.

#### Parameters

|                     |                                    |
|---------------------|------------------------------------|
| <code>__l</code>    | An <code>initializer_list</code> . |
| <code>__comp</code> | A comparison functor.              |
| <code>__a</code>    | An allocator object.               |

Create a multiset consisting of copies of the elements from the list. This is linear in  $N$  if the list is already sorted, and  $N \log N$  otherwise (where  $N$  is `__l.size()`).

Definition at line 214 of file `stl_multiset.h`.

### 4.882.3 Member Function Documentation

```
4.882.3.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 295 of file `stl_multiset.h`.

**4.882.3.2** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 332 of file stl\_multiset.h.

**4.882.3.3** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 341 of file stl\_multiset.h.

**4.882.3.4** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::multiset< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]`

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 614 of file stl\_multiset.h.

Referenced by std::multiset< \_Key, \_Compare, \_Alloc >::operator=().

**4.882.3.5** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type  
std::multiset< _Key, _Compare, _Alloc >::count ( const key_type & __x ) const [inline]`

Finds the number of elements with given key.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

#### Returns

Number of elements with specified key.

Definition at line 625 of file stl\_multiset.h.

**4.882.3.6** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 350 of file stl\_multiset.h.

**4.882.3.7** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 359 of file stl\_multiset.h.

```
4.882.3.8 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args> iterator std::multiset< _Key, _Compare, _Alloc >::emplace (_Args &&... _args)
[inline]
```

Builds and inserts an element into the multiset.

#### Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate the element instance to be inserted. |
|---------------------|-----------------------------------------------------------------|

#### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 409 of file `stl_multiset.h`.

```
4.882.3.9 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args> iterator std::multiset< _Key, _Compare, _Alloc >::emplace_hint (const_iterator _pos,
_Args &&... _args) [inline]
```

Builds and inserts an element into the multiset.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element instance to be inserted.               |

#### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 435 of file `stl_multiset.h`.

```
4.882.3.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool
std::multiset< _Key, _Compare, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 365 of file `stl_multiset.h`.

4.882.3.11 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset<_Key, _Compare, _Alloc>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 304 of file `stl_multiset.h`.

4.882.3.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::pair<iterator, iterator> std::multiset<_Key, _Compare, _Alloc>::equal_range ( const key_type & __x )  
[inline]`

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 705 of file `stl_multiset.h`.

4.882.3.13 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::pair<const_iterator, const_iterator> std::multiset<_Key, _Compare, _Alloc>::equal_range ( const key_type &  
__x ) const [inline]`

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 709 of file `stl_multiset.h`.

4.882.3.14 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::erase ( const_iterator __position ) [inline]`

Erases an element from a multiset.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 536 of file `stl_multiset.h`.

4.882.3.15 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type  
std::multiset< _Key, _Compare, _Alloc >::erase ( const key_type & __x ) [inline]`

Erases elements according to the provided key.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

#### Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 566 of file `stl_multiset.h`.

4.882.3.16 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::erase ( const_iterator __first, const_iterator __last ) [inline]`

Erases a `[first,last)` range of elements from a multiset.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

#### Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 587 of file stl\_multiset.h.

**4.882.3.17** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::find ( const key_type & __x ) [inline]`

Tries to locate an element in a set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 643 of file stl\_multiset.h.

**4.882.3.18** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::multiset< _Key, _Compare, _Alloc >::find ( const key_type & __x ) const [inline]`

Tries to locate an element in a set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 647 of file stl\_multiset.h.

**4.882.3.19** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]`

Returns the memory allocation object.

Definition at line 286 of file stl\_multiset.h.

**4.882.3.20** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::insert ( const value_type & __x ) [inline]`

Inserts an element into the multiset.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 454 of file stl\_multiset.h.

Referenced by std::multiset< \_Key, \_Compare, \_Alloc >::operator=().

**4.882.3.21** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x ) [inline]`

Inserts an element into the multiset.

**Parameters**

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 484 of file stl\_multiset.h.

**4.882.3.22** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename _InputIterator > void std::multiset< _Key, _Compare, _Alloc >::insert ( _InputIterator __first,  
_InputIterator __last ) [inline]`

A template function that tries to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 503 of file stl\_multiset.h.

**4.882.3.23** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::multiset< _Key, _Compare, _Alloc >::insert ( initializer_list< value_type > __l ) [inline]`

Attempts to insert a list of elements into the multiset.

## Parameters

|     |                                                                 |
|-----|-----------------------------------------------------------------|
| __l | A std::initializer_list<value_type> of elements to be inserted. |
|-----|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 515 of file stl\_multiset.h.

References std::multiset< \_Key, \_Compare, \_Alloc >::insert().

Referenced by std::multiset< \_Key, \_Compare, \_Alloc >::insert().

**4.882.3.24** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp ( ) const [inline]`

Returns the comparison object.

Definition at line 278 of file stl\_multiset.h.

**4.882.3.25** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::lower_bound ( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

## Parameters

|     |                    |
|-----|--------------------|
| __x | Key to be located. |
|-----|--------------------|

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 664 of file stl\_multiset.h.

**4.882.3.26** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound ( const key_type & __x ) const [inline]`

Finds the beginning of a subsequence matching given key.

## Parameters

|     |                    |
|-----|--------------------|
| __x | Key to be located. |
|-----|--------------------|

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 668 of file stl\_multiset.h.

**4.882.3.27** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type  
std::multiset< _Key, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]`

Returns the maximum size of the set.

Definition at line 375 of file stl\_multiset.h.

```
4.882.3.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (const multiset< _Key, _Compare, _Alloc > & __x)
 [inline]
```

Multiset assignment operator.

#### Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multiset of identical element and allocator types. |
|------------------|------------------------------------------------------|

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 229 of file stl\_multiset.h.

```
4.882.3.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (multiset< _Key, _Compare, _Alloc > && __x)
 [inline]
```

Multiset move assignment operator.

#### Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__x</code> | A multiset of identical element and allocator types. |
|------------------|------------------------------------------------------|

The contents of `__x` are moved into this multiset (without copying). `__x` is a valid, but unspecified multiset.

Definition at line 245 of file stl\_multiset.h.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::swap()`.

```
4.882.3.30 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (initializer_list< value_type > __l)
 [inline]
```

Multiset list assignment operator.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

This function fills a multiset with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 266 of file stl\_multiset.h.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::insert()`.

```
4.882.3.31 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 313 of file stl\_multiset.h.

4.882.3.32 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 322 of file stl\_multiset.h.

4.882.3.33 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type  
std::multiset< _Key, _Compare, _Alloc >::size ( ) const [inline], [noexcept]`

Returns the size of the set.

Definition at line 370 of file stl\_multiset.h.

4.882.3.34 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::multiset< _Key, _Compare, _Alloc >::swap ( multiset< _Key, _Compare, _Alloc > & _x ) [inline]`

Swaps data with another multiset.

#### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__x</code> | A multiset of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 390 of file stl\_multiset.h.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::operator=()`, and `std::swap()`.

4.882.3.35 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::multiset< _Key, _Compare, _Alloc >::upper_bound ( const key_type & _x ) [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 680 of file stl\_multiset.h.

4.882.3.36 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound ( const key_type & _x ) const [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 684 of file stl\_multiset.h.

4.882.3.37 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
value_compare std::multiset<_Key, _Compare, _Alloc>::value_comp ( ) const [inline]`

Returns the comparison object.

Definition at line 282 of file stl\_multiset.h.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

## 4.883 std::mutex Class Reference

Inherits std::\_\_mutex\_base.

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- `mutex (const mutex &)=delete`
- `void lock ()`
- `native_handle_type native_handle ()`
- `mutex & operator= (const mutex &)=delete`
- `bool try_lock () noexcept`
- `void unlock ()`

### Private Types

- `typedef __pthread_mutex_t __native_type`

### Private Attributes

- `__native_type _M_mutex`

### 4.883.1 Detailed Description

mutex

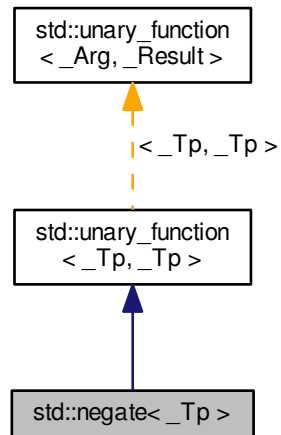
Definition at line 116 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

4.884 `std::negate<_Tp>` Struct Template Reference

Inheritance diagram for `std::negate<_Tp>`:



## Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Tp` [result\\_type](#)

## Public Member Functions

- `_Tp operator()` (`const _Tp &__x`) `const`

## 4.884.1 Detailed Description

`template<typename _Tp> struct std::negate<_Tp>`

One of the [math functors](#).

Definition at line 185 of file `stl_function.h`.

## 4.884.2 Member Typedef Documentation

4.884.2.1 typedef `_Tp` `std::unary_function<_Tp, _Tp>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.884.2.2 `typedef _Tp std::unary_function<_Tp, _Tp>::result_type` [inherited]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.885 `std::negative_binomial_distribution<_IntType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- `typedef _IntType` [result\\_type](#)

### Public Member Functions

- **`negative_binomial_distribution`** (`_IntType __k=1`, `double __p=0.5`)
- **`negative_binomial_distribution`** (`const` [param\\_type](#) &`__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void` **`__generate`** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void` **`__generate`** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, `const` [param\\_type](#) &`__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`void` **`__generate`** ([result\\_type](#) \* `__f`, [result\\_type](#) \* `__t`, `_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`void` **`__generate`** ([result\\_type](#) \* `__f`, [result\\_type](#) \* `__t`, `_UniformRandomNumberGenerator &__urng`, `const` [param\\_type](#) &`__p`)
- `_IntType` **`k`** () `const`
- [result\\_type](#) **`max`** () `const`
- [result\\_type](#) **`min`** () `const`
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`, `const` [param\\_type](#) &`__p`)
- `double` **`p`** () `const`
- [param\\_type](#) **`param`** () `const`
- `void` **`param`** (`const` [param\\_type](#) &`__param`)
- `void` **`reset`** ()

### Friends

- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT,`  
`_Traits > &` **`operator<<`** (`std::basic_ostream<_CharT, _Traits > &__os`, `const` `std::negative_binomial_`  
`distribution<_IntType1 > &__x`)

- bool operator==(const negative\_binomial\_distribution &\_\_d1, const negative\_binomial\_distribution &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
std::basic\_istream<\_CharT,  
\_Traits> & operator>> (std::basic\_istream<\_CharT, \_Traits> &\_\_is, std::negative\_binomial\_distribution<\_IntType1> &\_\_x)

#### 4.885.1 Detailed Description

template<typename \_IntType = int> class std::negative\_binomial\_distribution<\_IntType>

A negative\_binomial\_distribution random number distribution.

The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 4207 of file random.h.

#### 4.885.2 Member Typedef Documentation

4.885.2.1 template<typename \_IntType = int> typedef \_IntType std::negative\_binomial\_distribution<\_IntType>::result\_type

The type of the range of the distribution.

Definition at line 4210 of file random.h.

#### 4.885.3 Member Function Documentation

4.885.3.1 template<typename \_IntType = int> \_IntType std::negative\_binomial\_distribution<\_IntType>::k( ) const  
[inline]

Return the  $k$  parameter of the distribution.

Definition at line 4265 of file random.h.

4.885.3.2 template<typename \_IntType = int> result\_type std::negative\_binomial\_distribution<\_IntType>::max( )  
const [inline]

Returns the least upper bound value of the distribution.

Definition at line 4301 of file random.h.

References std::numeric\_limits<\_Tp>::max().

4.885.3.3 template<typename \_IntType = int> result\_type std::negative\_binomial\_distribution<\_IntType>::min( )  
const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4294 of file random.h.

4.885.3.4 template<typename \_IntType> template<typename UniformRandomNumberGenerator>  
negative\_binomial\_distribution<\_IntType>::result\_type std::negative\_binomial\_distribution<\_IntType>::operator()(UniformRandomNumberGenerator & \_\_urng )

Generating functions.

Definition at line 1278 of file bits/random.tcc.

4.885.3.5 `template<typename _IntType = int> double std::negative_binomial_distribution< _IntType >::p ( ) const`  
`[inline]`

Return the  $p$  parameter of the distribution.

Definition at line 4272 of file random.h.

4.885.3.6 `template<typename _IntType = int> param_type std::negative_binomial_distribution< _IntType >::param ( )`  
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 4279 of file random.h.

4.885.3.7 `template<typename _IntType = int> void std::negative_binomial_distribution< _IntType >::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4287 of file random.h.

4.885.3.8 `template<typename _IntType = int> void std::negative_binomial_distribution< _IntType >::reset ( )`  
`[inline]`

Resets the distribution state.

Definition at line 4258 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

#### 4.885.4 Friends And Related Function Documentation

4.885.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits> & operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const`  
`std::negative_binomial_distribution< _IntType1 > & __x ) [friend]`

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                           |
|-------------------|---------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                         |
| <code>__x</code>  | A <code>negative_binomial_distribution</code> random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

4.885.4.2 `template<typename _IntType = int> bool operator==( const negative_binomial_distribution<_IntType> & __d1, const negative_binomial_distribution<_IntType> & __d2 ) [friend]`

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4350 of file `random.h`.

4.885.4.3 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::negative_binomial_distribution<_IntType1> & __x ) [friend]`

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                              |
| <code>__x</code>  | A <code>negative_binomial_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.886 `std::negative_binomial_distribution<_IntType>::param_type` Struct Reference

## Public Types

- typedef  
[negative\\_binomial\\_distribution](#)  
<\_IntType> **distribution\_type**

## Public Member Functions

- **param\_type** (`_IntType` `__k`=1, `double` `__p`=0.5)
- `_IntType` **k** () const
- `double` **p** () const

## Friends

- `bool` **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.886.1 Detailed Description

```
template<typename _IntType = int>struct std::negative_binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4216 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.887 std::nested\_exception Class Reference

Inherited by std::\_Nested\_exception< \_Except >.

## Public Member Functions

- **nested\_exception** (const [nested\\_exception](#) &)=default
- exception\_ptr **nested\_ptr** () const
- [nested\\_exception](#) & **operator=** (const [nested\\_exception](#) &)=default
- void **rethrow\_nested** () const \_\_attribute\_\_((\_\_noreturn\_\_))

## 4.887.1 Detailed Description

Exception class with exception\_ptr data member.

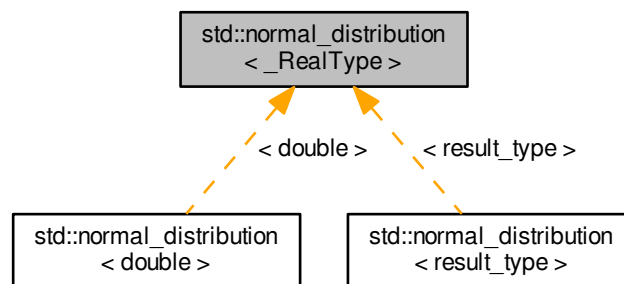
Definition at line 55 of file nested\_exception.h.

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

## 4.888 std::normal\_distribution&lt; \_RealType &gt; Class Template Reference

Inheritance diagram for std::normal\_distribution< \_RealType >:



## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- [normal\\_distribution](#) ([result\\_type](#) \_\_mean=[result\\_type](#)(0), [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- [normal\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) [max](#) () const
- \_RealType [mean](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()
- \_RealType [stddev](#) () const

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)<\_CharT, \_Traits> & [operator<<](#) ([std::basic\\_ostream](#)<\_CharT, \_Traits> &\_\_os, const [std::normal\\_distribution](#)<\_RealType1> &\_\_x)
- template<typename \_RealType1 >  
bool [operator==](#) (const [std::normal\\_distribution](#)<\_RealType1> &\_\_d1, const [std::normal\\_distribution](#)<\_RealType1> &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)<\_CharT, \_Traits> & [operator>>](#) ([std::basic\\_istream](#)<\_CharT, \_Traits> &\_\_is, [std::normal\\_distribution](#)<\_RealType1> &\_\_x)

## 4.888.1 Detailed Description

```
template<typename _RealType = double> class std::normal_distribution< _RealType >
```

A normal continuous distribution for random numbers.

The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}}$$

Definition at line 2085 of file random.h.

## 4.888.2 Member Typedef Documentation

```
4.888.2.1 template<typename _RealType = double> typedef _RealType std::normal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2088 of file random.h.

## 4.888.3 Constructor &amp; Destructor Documentation

```
4.888.3.1 template<typename _RealType = double> std::normal_distribution< _RealType >::normal_distribution
 (result_type __mean = result_type(0), result_type __stddev = result_type(1)) [inline],
 [explicit]
```

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2130 of file random.h.

## 4.888.4 Member Function Documentation

```
4.888.4.1 template<typename _RealType = double> result_type std::normal_distribution< _RealType >::max () const
 [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2187 of file random.h.

```
4.888.4.2 template<typename _RealType = double> _RealType std::normal_distribution< _RealType >::mean () const
 [inline]
```

Returns the mean of the distribution.

Definition at line 2151 of file random.h.

```
4.888.4.3 template<typename _RealType = double> result_type std::normal_distribution< _RealType >::min () const
 [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2180 of file random.h.

4.888.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type  
std::normal_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2195 of file random.h.

Referenced by std::normal\_distribution<result\_type>::operator()().

4.888.4.5 `template<typename _RealType> template<typename _UniformRandomNumberGenerator> normal_distribution<  
_RealType>::result_type std::normal_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator  
& __urng, const param_type & __param )`

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1933 of file bits/random.tcc.

References std::log(), and std::sqrt().

4.888.4.6 `template<typename _RealType = double> param_type std::normal_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2165 of file random.h.

4.888.4.7 `template<typename _RealType = double> void std::normal_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2173 of file random.h.

4.888.4.8 `template<typename _RealType = double> void std::normal_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2144 of file random.h.

Referenced by std::lognormal\_distribution<\_RealType>::reset(), std::gamma\_distribution<result\_type>::reset(), std::student\_t\_distribution<\_RealType>::reset(), std::binomial\_distribution<\_IntType>::reset(), and std::poisson\_distribution<\_IntType>::reset().

4.888.4.9 `template<typename _RealType = double> _RealType std::normal_distribution<_RealType>::stddev ( ) const  
[inline]`

Returns the standard deviation of the distribution.

Definition at line 2158 of file random.h.

#### 4.888.5 Friends And Related Function Documentation

4.888.5.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const  
std::normal_distribution<_RealType1> & __x ) [friend]`

Inserts a normal\_distribution random number distribution \_\_x into the output stream \_\_os.

#### Parameters

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <code>__os</code> | An output stream.                                 |
| <code>__x</code>  | A normal_distribution random number distribution. |

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

4.888.5.2 `template<typename _RealType = double> template<typename _RealType1> bool operator== ( const  
std::normal_distribution<_RealType1> & __d1, const std::normal_distribution<_RealType1> & __d2 )  
[friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

4.888.5.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits>  
> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is,  
std::normal_distribution<_RealType1> & __x ) [friend]`

Extracts a normal\_distribution random number distribution \_\_x from the input stream \_\_is.

#### Parameters

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <code>__is</code> | An input stream.                                      |
| <code>__x</code>  | A normal_distribution random number generator engine. |

#### Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.889 std::normal\_distribution<\_RealType>::param\_type Struct Reference

#### Public Types

- typedef [normal\\_distribution](#)  
<\_RealType> **distribution\_type**

#### Public Member Functions

- **param\_type** (\_RealType \_\_mean=\_RealType(0), \_RealType \_\_stddev=\_RealType(1))
- \_RealType **mean** () const
- \_RealType **stddev** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.889.1 Detailed Description

template<typename [\\_RealType](#) = double>struct std::normal\_distribution< [\\_RealType](#) >::param\_type

Parameter type.

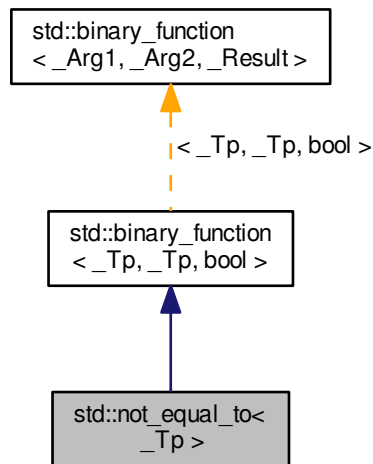
Definition at line 2094 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.890 std::not\_equal\_to&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::not\_equal\_to< \_Tp >:



## Public Types

- typedef [\\_Tp](#) [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef [\\_Tp](#) [second\\_argument\\_type](#)

## Public Member Functions

- bool **operator()** (const [\\_Tp](#) &\_\_x, const [\\_Tp](#) &\_\_y) const

## 4.890.1 Detailed Description

```
template<typename _Tp>struct std::not_equal_to<_Tp >
```

One of the [comparison functors](#).

Definition at line 213 of file stl\_function.h.

## 4.890.2 Member Typedef Documentation

4.890.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl\_function.h.

4.890.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool >::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 123 of file stl\_function.h.

4.890.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool >::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

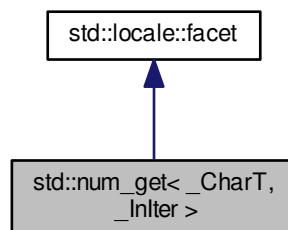
Definition at line 120 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.891 std::num\_get&lt;\_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for `std::num_get<_CharT, _InIter >`:



## Public Types

- `typedef _CharT` [char\\_type](#)

- typedef `_InIter` `iter_type`

#### Public Member Functions

- `num_get` (`size_t __refs=0`)
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, void *& __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v`) const
- `iter_type get` (`iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v`) const

#### Static Public Attributes

- static `locale::id` `id`

#### Protected Member Functions

- virtual `~num_get` ()
- `iter_type _M_extract_float` (`iter_type, iter_type, ios_base &, ios_base::iostate &, string &`) const
- `template<typename _ValueT>`  
`iter_type _M_extract_int` (`iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &`) const
- `template<typename _CharT2>`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT2 >`  
`::value, int >::__type` `_M_find` (`const _CharT2 *, size_t __len, _CharT2 __c`) const
- `template<typename _CharT2>`  
`__gnu_cxx::__enable_if`  
`< !__is_char< _CharT2 >`  
`::value, int >::__type` `_M_find` (`const _CharT2 * __zero, size_t __len, _CharT2 __c`) const
- virtual `iter_type do_get` (`iter_type, iter_type, ios_base &, ios_base::iostate &, bool &`) const
- virtual `iter_type do_get` (`iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v`) const
- virtual `iter_type do_get` (`iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v`) const
- virtual `iter_type do_get` (`iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v`) const
- virtual `iter_type do_get` (`iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v`) const

- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const`

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### 4.891.1 Detailed Description

`template<typename _CharT, typename _InIter>class std::num_get< _CharT, _InIter >`

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.

The `num_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_get` facet.

Definition at line 1915 of file `locale_facets.h`.

#### 4.891.2 Member Typedef Documentation

4.891.2.1 `template<typename _CharT, typename _InIter > typedef _CharT std::num_get< _CharT, _InIter >::char_type`

Public typedefs.

Definition at line 1921 of file `locale_facets.h`.

4.891.2.2 `template<typename _CharT, typename _InIter > typedef _InIter std::num_get< _CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1922 of file `locale_facets.h`.

#### 4.891.3 Constructor & Destructor Documentation

4.891.3.1 `template<typename _CharT, typename _InIter > std::num_get< _CharT, _InIter >::num_get ( size_t __refs = 0 )  
[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1936 of file locale\_facets.h.

4.891.3.2 `template<typename _CharT, typename _Inlter > virtual std::num_get<_CharT, _Inlter >::~num_get ( )`  
`[inline], [protected], [virtual]`

Destructor.

Definition at line 2108 of file locale\_facets.h.

## 4.891.4 Member Function Documentation

4.891.4.1 `template<typename _CharT, typename _Inlter > _Inlter std::num_get<_CharT, _Inlter >::do_get ( iter_type __beg,`  
`iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v ) const` `[protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See Also

`get()` for more details.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 590 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

4.891.4.2 `template<typename _CharT, typename _Inlter > virtual iter_type std::num_get<_CharT, _Inlter >::do_get (`  
`iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v ) const` `[inline],`  
`[protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See Also

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2176 of file locale\_facets.h.

```
4.891.4.3 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v) const
 [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See Also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2181 of file locale\_facets.h.

```
4.891.4.4 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const
 [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See Also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2186 of file locale\_facets.h.

```
4.891.4.5 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const
 [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See Also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2191 of file locale\_facets.h.

```
4.891.4.6 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const
 [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See Also**

`get()` for more details.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 2197 of file locale\_facets.h.

```
4.891.4.7 template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const
 [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See Also

`get()` for more details.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 2202 of file locale\_facets.h.

```
4.891.4.8 template<typename _CharT, typename _InIter > _InIter std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
 iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See Also

`get()` for more details.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 686 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

4.891.4.9 `template<typename _CharT, typename _Inlter > _Inlter std::num_get< _CharT, _Inlter >::do_get ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v ) const [protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See Also**

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 701 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

4.891.4.10 `template<typename _CharT, typename _Inlter > _Inlter std::num_get< _CharT, _Inlter >::do_get ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v ) const [protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See Also**

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 733 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

**4.891.4.11** `template<typename _CharT, typename _Inlter > _Inlter std::num_get< _CharT, _Inlter >::do_get ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, void *& __v ) const [protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

**See Also**

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 748 of file locale\_facets.tcc.

References std::ios\_base::basefield, std::ios\_base::flags(), and std::ios\_base::hex.

**4.891.4.12** `template<typename _CharT, typename _Inlter > iter_type std::num_get< _CharT, _Inlter >::get ( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the bool v. It does so by calling num\_get::do\_get().

If ios\_base::boolalpha is set, attempts to read ctype<CharT>::truenamename() or ctype<CharT>::falsenamename(). Sets v to true or false if successful. Sets err to ios\_base::failbit if reading the string fails. Sets err to ios\_base::eofbit if the stream is emptied.

If ios\_base::boolalpha is not set, proceeds as with reading a long, except if the value is 1, sets v to true, if the value is 0, sets v to false, and otherwise set err to ios\_base::failbit.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 1962 of file locale\_facets.h.

Referenced by std::basic\_istream<\_CharT, \_Traits>::operator>>().

4.891.4.13 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 1999 of file locale\_facets.h.

4.891.4.14 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 2004 of file locale\_facets.h.

```
4.891.4.15 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get (iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 2009 of file locale\_facets.h.

```
4.891.4.16 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get (iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2014 of file `locale_facets.h`.

**4.891.4.17** `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2020 of file `locale_facets.h`.

**4.891.4.18** `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2025 of file `locale_facets.h`.

**4.891.4.19** `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf` `g` specifier. The matching type length modifier is also used.

The decimal point character used is `numpunct::decimal_point()`. Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2059 of file `locale_facets.h`.

**4.891.4.20** `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get( iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2064 of file `locale_facets.h`.

```
4.891.4.21 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get(iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2069 of file `locale_facets.h`.

```
4.891.4.22 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get(iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, void *& __v) const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the scanf p specifier.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2102 of file locale\_facets.h.

#### 4.891.5 Member Data Documentation

4.891.5.1 `template<typename _CharT, typename _Inlter > locale::id std::num_get< _CharT, _Inlter >::id` `[static]`

Numpunct facet id.

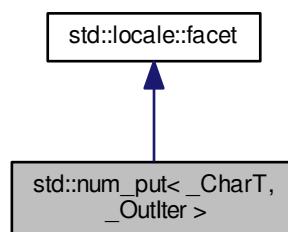
Definition at line 1926 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

## 4.892 std::num\_put< \_CharT, \_Outlter > Class Template Reference

Inheritance diagram for std::num\_put< \_CharT, \_Outlter >:



## Public Types

- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_OutIter](#) [iter\\_type](#)

## Public Member Functions

- [num\\_put](#) (size\_t \_\_refs=0)
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, bool \_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const void \*\_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long \_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long \_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long long \_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long long \_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, double \_\_v) const
- [iter\\_type put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long double \_\_v) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~num\\_put](#) ()
- void [\\_M\\_group\\_float](#) (const char \*\_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, const char\_type \*\_\_p, char\_type \*\_\_new, char\_type \*\_\_cs, int & \_\_len) const
- void [\\_M\\_group\\_int](#) (const char \*\_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, ios\_base & \_\_io, char\_type \*\_\_new, char\_type \*\_\_cs, int & \_\_len) const
- template<typename \_ValueT >  
[iter\\_type \\_M\\_insert\\_float](#) (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, char \_\_mod, \_ValueT \_\_v) const
- template<typename \_ValueT >  
[iter\\_type \\_M\\_insert\\_int](#) (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, \_ValueT \_\_v) const
- void [\\_M\\_pad](#) (char\_type \_\_fill, streamsize \_\_w, ios\_base & \_\_io, char\_type \*\_\_new, const char\_type \*\_\_cs, int & \_\_len) const
- virtual [iter\\_type do\\_put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, bool \_\_v) const
- virtual [iter\\_type do\\_put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long \_\_v) const
- virtual [iter\\_type do\\_put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long \_\_v) const
- virtual [iter\\_type do\\_put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long long \_\_v) const
- virtual [iter\\_type do\\_put](#) (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long long \_\_v) const
- virtual [iter\\_type do\\_put](#) (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, double) const
- virtual [iter\\_type do\\_put](#) (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, long double) const
- virtual [iter\\_type do\\_put](#) (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, const void \*) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.892.1 Detailed Description

template<typename \_CharT, typename \_Outiter>class std::num\_put< \_CharT, \_Outiter >

Primary class template num\_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

The num\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_put facet.

Definition at line 2254 of file locale\_facets.h.

## 4.892.2 Member Typedef Documentation

4.892.2.1 template<typename \_CharT, typename \_Outiter > typedef \_CharT std::num\_put< \_CharT, \_Outiter >::char\_type

Public typedefs.

Definition at line 2260 of file locale\_facets.h.

4.892.2.2 template<typename \_CharT, typename \_Outiter > typedef \_Outiter std::num\_put< \_CharT, \_Outiter >::iter\_type

Public typedefs.

Definition at line 2261 of file locale\_facets.h.

## 4.892.3 Constructor &amp; Destructor Documentation

4.892.3.1 template<typename \_CharT, typename \_Outiter > std::num\_put< \_CharT, \_Outiter >::num\_put ( size\_t \_\_refs = 0 )  
[inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 2275 of file locale\_facets.h.

4.892.3.2 template<typename \_CharT, typename \_Outiter > virtual std::num\_put< \_CharT, \_Outiter >::~~num\_put ( )  
[inline], [protected], [virtual]

Destructor.

Definition at line 2454 of file locale\_facets.h.

#### 4.892.4 Member Function Documentation

4.892.4.1 `template<typename _CharT, typename _Outiter> _Outiter std::num_put<_CharT, _Outiter>::do_put ( iter_type __s, ios_base & __io, char_type __fill, bool __v ) const` `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

##### Parameters

|                     |                                            |
|---------------------|--------------------------------------------|
| <code>__s</code>    | Stream to write to.                        |
| <code>__io</code>   | Source of locale and flags.                |
| <code>__fill</code> | <code>Char_type</code> to use for filling. |
| <code>__v</code>    | Value to format and insert.                |

##### Returns

Iterator after writing.

Definition at line 1089 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::adjustfield`, `std::ios_base::boolalpha`, `std::ios_base::flags()`, `std::ios_base::left`, and `std::ios_base::width()`.

4.892.4.2 `template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put ( iter_type __s, ios_base & __io, char_type __fill, long __v ) const` `[inline]`, `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

##### Parameters

|                     |                                            |
|---------------------|--------------------------------------------|
| <code>__s</code>    | Stream to write to.                        |
| <code>__io</code>   | Source of locale and flags.                |
| <code>__fill</code> | <code>Char_type</code> to use for filling. |
| <code>__v</code>    | Value to format and insert.                |

##### Returns

Iterator after writing.

Definition at line 2474 of file locale\_facets.h.

4.892.4.3 `template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put ( iter_type __s, ios_base & __io, char_type __fill, unsigned long __v ) const` `[inline]`, `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2478 of file locale\_facets.h.

```
4.892.4.4 template<typename _CharT, typename _Outiter > virtual iter_type std::num_put< _CharT, _Outiter >::do_put (
 iter_type __s, ios_base & __io, char_type __fill, long long __v) const [inline], [protected],
 [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2484 of file locale\_facets.h.

```
4.892.4.5 template<typename _CharT, typename _Outiter > virtual iter_type std::num_put< _CharT, _Outiter >::do_put (
 iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v) const [inline], [protected],
 [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2489 of file locale\_facets.h.

4.892.4.6 `template<typename _CharT, typename _Outiter > _Outiter std::num_put<_CharT, _Outiter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, double __v ) const` [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

Definition at line 1141 of file locale\_facets.tcc.

4.892.4.7 `template<typename _CharT, typename _Outiter > _Outiter std::num_put<_CharT, _Outiter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, long double __v ) const` [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

Definition at line 1155 of file locale\_facets.tcc.

4.892.4.8 `template<typename _CharT, typename _Outiter > _Outiter std::num_put<_CharT, _Outiter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, const void * __v ) const` [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 1162 of file locale\_facets.tcc.

References std::ios\_base::flags(), std::ios\_base::hex, and std::ios\_base::uppercase.

4.892.4.9 `template<typename _CharT, typename _Outiter> iter_type std::num_put<_CharT, _Outiter>::put ( iter_type __s, ios_base & __io, char_type __fill, bool __v ) const [inline]`

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

If ios\_base::boolalpha is set, writes ctype<CharT>::truenamex() or ctype<CharT>::falsenamex(). Otherwise formats *v* as an int.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 2293 of file locale\_facets.h.

4.892.4.10 `template<typename _CharT, typename _Outiter> iter_type std::num_put<_CharT, _Outiter>::put ( iter_type __s, ios_base & __io, char_type __fill, long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, formats like the printf o specifier. Else if equal to ios\_base::hex, formats like x or X with ios\_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios\_base::showpos is set, '+' is output before positive values. If ios\_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal\_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands\_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios\_base::adjustfield) == ios\_base::left, result is padded at the end. If ios\_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 2335 of file locale\_facets.h.

**4.892.4.11** `template<typename _CharT, typename _Outiter> iter_type std::num_put<_CharT, _Outiter>::put ( iter_type __s, ios_base & __io, char_type __fill, unsigned long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, formats like the printf o specifier. Else if equal to ios\_base::hex, formats like x or X with ios\_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios\_base::showpos is set, '+' is output before positive values. If ios\_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal\_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands\_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios\_base::adjustfield) == ios\_base::left, result is padded at the end. If ios\_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 2339 of file locale\_facets.h.

**4.892.4.12** `template<typename _CharT, typename _Outiter> iter_type std::num_put<_CharT, _Outiter>::put ( iter_type __s, ios_base & __io, char_type __fill, long long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, formats like the printf o specifier. Else if equal to ios\_base::hex, formats like x or X with ios\_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios\_base::showpos is set, '+' is output before positive values. If ios\_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal\_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands\_sep().

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

Definition at line 2345 of file `locale_facets.h`.

**4.892.4.13** `template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put ( iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v ) const [inline]`

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

Definition at line 2349 of file `locale_facets.h`.

**4.892.4.14** `template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put ( iter_type __s, ios_base & __io, char_type __fill, double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

#### Parameters

|                     |                                            |
|---------------------|--------------------------------------------|
| <code>__s</code>    | Stream to write to.                        |
| <code>__io</code>   | Source of locale and flags.                |
| <code>__fill</code> | <code>Char_type</code> to use for filling. |
| <code>__v</code>    | Value to format and insert.                |

#### Returns

Iterator after writing.

Definition at line 2398 of file `locale_facets.h`.

**4.892.4.15** `template<typename _CharT, typename _OutIter> iter_type std::num_put<_CharT, _OutIter>::put ( iter_type __s, ios_base & __io, char_type __fill, long double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2402 of file locale\_facets.h.

```
4.892.4.16 template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put (iter_type
__s, ios_base & __io, char_type __fill, const void * __v) const [inline]
```

Numeric formatting.

Formats the pointer value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats `v` as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2423 of file locale\_facets.h.

## 4.892.5 Member Data Documentation

```
4.892.5.1 template<typename _CharT, typename _Outiter > locale::id std::num_put<_CharT, _Outiter >::id [static]
```

Numpunct facet id.

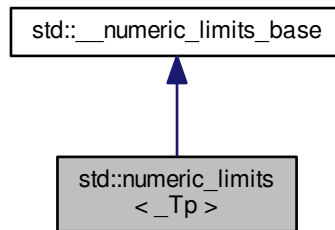
Definition at line 2265 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

4.893 `std::numeric_limits<_Tp>` Struct Template Reference

Inheritance diagram for `std::numeric_limits<_Tp>`:



## Static Public Member Functions

- static constexpr `_Tp` [denorm\\_min](#) () noexcept
- static constexpr `_Tp` [epsilon](#) () noexcept
- static constexpr `_Tp` [infinity](#) () noexcept
- static constexpr `_Tp` [lowest](#) () noexcept
- static constexpr `_Tp` [max](#) () noexcept
- static constexpr `_Tp` [min](#) () noexcept
- static constexpr `_Tp` [quiet\\_NaN](#) () noexcept
- static constexpr `_Tp` [round\\_error](#) () noexcept
- static constexpr `_Tp` [signaling\\_NaN](#) () noexcept

## Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)

- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

#### 4.893.1 Detailed Description

template<typename \_Tp>struct std::numeric\_limits< \_Tp >

Properties of fundamental types.

This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS: DRs 201 and 184 (hi Gaby!) are noted, but not incorporated in this documented (yet).

Definition at line 304 of file limits.

#### 4.893.2 Member Function Documentation

4.893.2.1 template<typename \_Tp > static constexpr \_Tp std::numeric\_limits< \_Tp >::denorm\_min ( ) [inline],  
[static], [noexcept]

The minimum positive denormalized value. For types where `has_denorm` is `false`, this is the minimum positive normalized value.

Definition at line 349 of file limits.

4.893.2.2 template<typename \_Tp > static constexpr \_Tp std::numeric\_limits< \_Tp >::epsilon ( ) [inline],  
[static], [noexcept]

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 325 of file limits.

Referenced by `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

4.893.2.3 template<typename \_Tp > static constexpr \_Tp std::numeric\_limits< \_Tp >::infinity ( ) [inline],  
[static], [noexcept]

The representation of positive infinity, if `has_infinity`.

Definition at line 333 of file limits.

4.893.2.4 template<typename \_Tp > static constexpr \_Tp std::numeric\_limits< \_Tp >::lowest ( ) [inline],  
[static], [noexcept]

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 319 of file limits.

**4.893.2.5** `template<typename _Tp> static constexpr _Tp std::numeric_limits< _Tp >::max ( ) [inline], [static], [noexcept]`

The maximum finite value.

Definition at line 313 of file limits.

Referenced by `std::normal_distribution< result_type >::max()`, `std::lognormal_distribution< _RealType >::max()`, `std::gamma_distribution< result_type >::max()`, `std::chi_squared_distribution< _RealType >::max()`, `std::cauchy_distribution< _RealType >::max()`, `std::fisher_f_distribution< _RealType >::max()`, `std::student_t_distribution< _RealType >::max()`, `std::bernoulli_distribution::max()`, `std::geometric_distribution< _IntType >::max()`, `std::negative_binomial_distribution< _IntType >::max()`, `std::poisson_distribution< _IntType >::max()`, `std::exponential_distribution< _RealType >::max()`, `std::weibull_distribution< _RealType >::max()`, `std::extreme_value_distribution< _RealType >::max()`, `std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size()`, `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()`, `std::binomial_distribution< _IntType >::operator()`, and `std::poisson_distribution< _IntType >::operator()`.

**4.893.2.6** `template<typename _Tp> static constexpr _Tp std::numeric_limits< _Tp >::min ( ) [inline], [static], [noexcept]`

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 309 of file limits.

Referenced by `std::normal_distribution< result_type >::min()`, `std::cauchy_distribution< _RealType >::min()`, `std::student_t_distribution< _RealType >::min()`, `std::bernoulli_distribution::min()`, `std::extreme_value_distribution< _RealType >::min()`, and `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()`.

**4.893.2.7** `template<typename _Tp> static constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN ( ) [inline], [static], [noexcept]`

The representation of a quiet Not a Number, if `has_quiet_NaN`.

Definition at line 338 of file limits.

**4.893.2.8** `template<typename _Tp> static constexpr _Tp std::numeric_limits< _Tp >::round_error ( ) [inline], [static], [noexcept]`

The maximum rounding error measurement (see LIA-1).

Definition at line 329 of file limits.

**4.893.2.9** `template<typename _Tp> static constexpr _Tp std::numeric_limits< _Tp >::signaling_NaN ( ) [inline], [static], [noexcept]`

The representation of a signaling Not a Number, if `has_signaling_NaN`.

Definition at line 343 of file limits.

### 4.893.3 Member Data Documentation

**4.893.3.1** `constexpr int std::_numeric_limits_base::digits [static], [inherited]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 200 of file limits.

**4.893.3.2** constexpr int std::numeric\_limits\_base::digits10 [static],[inherited]

The number of base 10 digits that can be represented without change.

Definition at line 203 of file limits.

**4.893.3.3** constexpr float\_denorm\_style std::numeric\_limits\_base::has\_denorm [static],[inherited]

See std::float\_denorm\_style for more information.

Definition at line 255 of file limits.

**4.893.3.4** constexpr bool std::numeric\_limits\_base::has\_denorm\_loss [static],[inherited]

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 259 of file limits.

**4.893.3.5** constexpr bool std::numeric\_limits\_base::has\_infinity [static],[inherited]

True if the type has a representation for positive infinity.

Definition at line 244 of file limits.

**4.893.3.6** constexpr bool std::numeric\_limits\_base::has\_quiet\_NaN [static],[inherited]

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 248 of file limits.

**4.893.3.7** constexpr bool std::numeric\_limits\_base::has\_signaling\_NaN [static],[inherited]

True if the type has a representation for a signaling Not a Number.

Definition at line 252 of file limits.

**4.893.3.8** constexpr bool std::numeric\_limits\_base::is\_bounded [static],[inherited]

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 268 of file limits.

**4.893.3.9** constexpr bool std::numeric\_limits\_base::is\_exact [static],[inherited]

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 220 of file limits.

**4.893.3.10** constexpr bool std::numeric\_limits\_base::is\_iec559 [static],[inherited]

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 263 of file limits.

**4.893.3.11** constexpr bool std::numeric\_limits\_base::is\_integer [static],[inherited]

True if the type is integer.

Definition at line 215 of file limits.

**4.893.3.12** constexpr bool std::numeric\_limits\_base::is\_modulo [static],[inherited]

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()), the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 277 of file limits.

**4.893.3.13** constexpr bool std::numeric\_limits\_base::is\_signed [static],[inherited]

True if the type is signed.

Definition at line 212 of file limits.

**4.893.3.14** constexpr bool std::numeric\_limits\_base::is\_specialized [static],[inherited]

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 195 of file limits.

**4.893.3.15** constexpr int std::numeric\_limits\_base::max\_digits10 [static],[inherited]

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 208 of file limits.

**4.893.3.16** constexpr int std::numeric\_limits\_base::max\_exponent [static],[inherited]

The maximum positive integer such that *radix* raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 237 of file limits.

**4.893.3.17** constexpr int std::numeric\_limits\_base::max\_exponent10 [static],[inherited]

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 241 of file limits.

**4.893.3.18** constexpr int std::numeric\_limits\_base::min\_exponent [static],[inherited]

The minimum negative integer such that *radix* raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 228 of file limits.

**4.893.3.19** constexpr int std::numeric\_limits\_base::min\_exponent10 [static],[inherited]

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 232 of file limits.

**4.893.3.20** constexpr int std::numeric\_limits\_base::radix [static],[inherited]

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 224 of file limits.

4.893.3.21 `constexpr float_round_style std::_numeric_limits_base::round_style` `[static], [inherited]`

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 288 of file `limits`.

4.893.3.22 `constexpr bool std::_numeric_limits_base::tinyness_before` `[static], [inherited]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 283 of file `limits`.

4.893.3.23 `constexpr bool std::_numeric_limits_base::traps` `[static], [inherited]`

True if trapping is implemented for this type.

Definition at line 280 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.894 `std::numeric_limits< bool >` Struct Template Reference

### Static Public Member Functions

- static `constexpr bool` **denorm\_min** () noexcept
- static `constexpr bool` **epsilon** () noexcept
- static `constexpr bool` **infinity** () noexcept
- static `constexpr bool` **lowest** () noexcept
- static `constexpr bool` **max** () noexcept
- static `constexpr bool` **min** () noexcept
- static `constexpr bool` **quiet\_NaN** () noexcept
- static `constexpr bool` **round\_error** () noexcept
- static `constexpr bool` **signaling\_NaN** () noexcept

### Static Public Attributes

- static `constexpr int` **digits**
- static `constexpr int` **digits10**
- static `constexpr float_denorm_style` **has\_denorm**
- static `constexpr bool` **has\_denorm\_loss**
- static `constexpr bool` **has\_infinity**
- static `constexpr bool` **has\_quiet\_NaN**
- static `constexpr bool` **has\_signaling\_NaN**
- static `constexpr bool` **is\_bounded**
- static `constexpr bool` **is\_exact**
- static `constexpr bool` **is\_iec559**
- static `constexpr bool` **is\_integer**
- static `constexpr bool` **is\_modulo**
- static `constexpr bool` **is\_signed**
- static `constexpr bool` **is\_specialized**
- static `constexpr int` **max\_digits10**

- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.894.1 Detailed Description

`template<> struct std::numeric_limits< bool >`

`numeric_limits<bool>` specialization.

Definition at line 371 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.895 `std::numeric_limits< char >` Struct Template Reference

### Static Public Member Functions

- static constexpr char **denorm\_min** () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char **max** () noexcept
- static constexpr char **min** () noexcept
- static constexpr char **quiet\_NaN** () noexcept
- static constexpr char **round\_error** () noexcept
- static constexpr char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**

- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.895.1 Detailed Description

`template<>struct std::numeric_limits< char >`

`numeric_limits<char>` specialization.

Definition at line 440 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.896 `std::numeric_limits< char16_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr `char16_t` **denorm\_min** () noexcept
- static constexpr `char16_t` **epsilon** () noexcept
- static constexpr `char16_t` **infinity** () noexcept
- static constexpr `char16_t` **lowest** () noexcept
- static constexpr `char16_t` **max** () noexcept
- static constexpr `char16_t` **min** () noexcept
- static constexpr `char16_t` **quiet\_NaN** () noexcept
- static constexpr `char16_t` **round\_error** () noexcept
- static constexpr `char16_t` **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**

- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.896.1 Detailed Description

`template<>struct std::numeric_limits< char16_t >`

`numeric_limits<char16_t>` specialization.

Definition at line 719 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.897 `std::numeric_limits< char32_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr `char32_t` **denorm\_min** () noexcept
- static constexpr `char32_t` **epsilon** () noexcept
- static constexpr `char32_t` **infinity** () noexcept
- static constexpr `char32_t` **lowest** () noexcept
- static constexpr `char32_t` **max** () noexcept
- static constexpr `char32_t` **min** () noexcept
- static constexpr `char32_t` **quiet\_NaN** () noexcept
- static constexpr `char32_t` **round\_error** () noexcept
- static constexpr `char32_t` **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**

- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.897.1 Detailed Description

`template<> struct std::numeric_limits< char32_t >`

`numeric_limits<char32_t>` specialization.

Definition at line 780 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.898 `std::numeric_limits< double >` Struct Template Reference

### Static Public Member Functions

- static constexpr double **denorm\_min** () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double **max** () noexcept
- static constexpr double **min** () noexcept
- static constexpr double **quiet\_NaN** () noexcept
- static constexpr double **round\_error** () noexcept
- static constexpr double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**

- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.898.1 Detailed Description

`template<>struct std::numeric_limits< double >`

`numeric_limits<double>` specialization.

Definition at line 1628 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.899 `std::numeric_limits< float >` Struct Template Reference

### Static Public Member Functions

- static constexpr float **denorm\_min** () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float **max** () noexcept
- static constexpr float **min** () noexcept
- static constexpr float **quiet\_NaN** () noexcept
- static constexpr float **round\_error** () noexcept
- static constexpr float **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**

- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.899.1 Detailed Description

`template<>struct std::numeric_limits< float >`

`numeric_limits<float>` specialization.

Definition at line 1553 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.900 `std::numeric_limits< int >` Struct Template Reference

### Static Public Member Functions

- static constexpr int **denorm\_min** () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int **max** () noexcept
- static constexpr int **min** () noexcept
- static constexpr int **quiet\_NaN** () noexcept
- static constexpr int **round\_error** () noexcept
- static constexpr int **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**

- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.900.1 Detailed Description

`template<>struct std::numeric_limits< int >`

`numeric_limits<int>` specialization.

Definition at line 982 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.901 `std::numeric_limits< long >` Struct Template Reference

### Static Public Member Functions

- static constexpr long **denorm\_min** () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long **max** () noexcept
- static constexpr long **min** () noexcept
- static constexpr long **quiet\_NaN** () noexcept
- static constexpr long **round\_error** () noexcept
- static constexpr long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**

- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.901.1 Detailed Description

`template<>struct std::numeric_limits< long >`

`numeric_limits<long>` specialization.

Definition at line 1121 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.902 `std::numeric_limits< long double >` Struct Template Reference

### Static Public Member Functions

- static constexpr long double **denorm\_min** () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double **min** () noexcept
- static constexpr long double **quiet\_NaN** () noexcept
- static constexpr long double **round\_error** () noexcept
- static constexpr long double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**

- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.902.1 Detailed Description

`template<>struct std::numeric_limits< long double >`

`numeric_limits<long double>` specialization.

Definition at line 1703 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.903 `std::numeric_limits< long long >` Struct Template Reference

### Static Public Member Functions

- static constexpr long long **denorm\_min** () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long **quiet\_NaN** () noexcept
- static constexpr long long **round\_error** () noexcept
- static constexpr long long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**

- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.903.1 Detailed Description

`template<>struct std::numeric_limits< long long >`

`numeric_limits<long long>` specialization.

Definition at line 1261 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.904 `std::numeric_limits< short >` Struct Template Reference

### Static Public Member Functions

- static constexpr short **denorm\_min** () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short **max** () noexcept
- static constexpr short **min** () noexcept
- static constexpr short **quiet\_NaN** () noexcept
- static constexpr short **round\_error** () noexcept
- static constexpr short **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**

- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 4.904.1 Detailed Description

`template<> struct std::numeric_limits< short >`

`numeric_limits<short>` specialization.

Definition at line 842 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.905 `std::numeric_limits< signed char >` Struct Template Reference

### Static Public Member Functions

- static constexpr signed char **denorm\_min** () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char **quiet\_NaN** () noexcept
- static constexpr signed char **round\_error** () noexcept
- static constexpr signed char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.905.1 Detailed Description

`template<> struct std::numeric_limits< signed char >`

`numeric_limits<signed char>` specialization.

Definition at line 507 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.906 `std::numeric_limits< unsigned char >` Struct Template Reference

## Static Public Member Functions

- static constexpr unsigned char **denorm\_min** () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char **min** () noexcept
- static constexpr unsigned char **quiet\_NaN** () noexcept
- static constexpr unsigned char **round\_error** () noexcept
- static constexpr unsigned char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.906.1 Detailed Description

`template<> struct std::numeric_limits< unsigned char >`

`numeric_limits<unsigned char>` specialization.

Definition at line 577 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.907 `std::numeric_limits< unsigned int >` Struct Template Reference

## Static Public Member Functions

- static constexpr unsigned int **denorm\_min** () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int **quiet\_NaN** () noexcept
- static constexpr unsigned int **round\_error** () noexcept
- static constexpr unsigned int **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.907.1 Detailed Description

`template<> struct std::numeric_limits< unsigned int >`

`numeric_limits<unsigned int>` specialization.

Definition at line 1049 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.908 `std::numeric_limits< unsigned long >` Struct Template Reference

## Static Public Member Functions

- static constexpr unsigned long **denorm\_min** () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long **quiet\_NaN** () noexcept
- static constexpr unsigned long **round\_error** () noexcept
- static constexpr unsigned long **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.908.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long >`

`numeric_limits<unsigned long>` specialization.

Definition at line 1188 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.909 `std::numeric_limits< unsigned long long >` Struct Template Reference

## Static Public Member Functions

- static constexpr unsigned long long **denorm\_min** () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long **quiet\_NaN** () noexcept
- static constexpr unsigned long long **round\_error** () noexcept
- static constexpr unsigned long long **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.909.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long long >`

`numeric_limits<unsigned long long>` specialization.

Definition at line 1331 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.910 `std::numeric_limits< unsigned short >` Struct Template Reference

## Static Public Member Functions

- static constexpr unsigned short **denorm\_min** () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short **min** () noexcept
- static constexpr unsigned short **quiet\_NaN** () noexcept
- static constexpr unsigned short **round\_error** () noexcept
- static constexpr unsigned short **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.910.1 Detailed Description

`template<> struct std::numeric_limits< unsigned short >`

`numeric_limits<unsigned short>` specialization.

Definition at line 909 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.911 `std::numeric_limits< wchar_t >` Struct Template Reference

## Static Public Member Functions

- static constexpr `wchar_t` **denorm\_min** () noexcept
- static constexpr `wchar_t` **epsilon** () noexcept
- static constexpr `wchar_t` **infinity** () noexcept
- static constexpr `wchar_t` **lowest** () noexcept
- static constexpr `wchar_t` **max** () noexcept
- static constexpr `wchar_t` **min** () noexcept
- static constexpr `wchar_t` **quiet\_NaN** () noexcept
- static constexpr `wchar_t` **round\_error** () noexcept
- static constexpr `wchar_t` **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.911.1 Detailed Description

`template<> struct std::numeric_limits< wchar_t >`

`numeric_limits<wchar_t>` specialization.

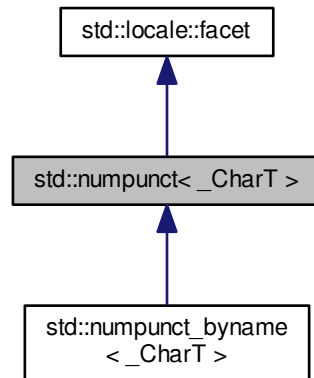
Definition at line 650 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.912 `std::num_punct<_CharT>` Class Template Reference

Inheritance diagram for `std::num_punct<_CharT>`:



## Public Types

- typedef `__num_punct_cache<_CharT>` `__cache_type`
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

## Public Member Functions

- `num_punct` (`size_t __refs=0`)
- `num_punct` (`__cache_type *__cache, size_t __refs=0`)
- `num_punct` (`__c_locale __cloc, size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~num_punct` ()
- void `_M_initialize_num_punct` (`__c_locale __cloc=0`)

- `template<>`  
`void _M_initialize_numpunct (__c_locale __cloc)`
- `template<>`  
`void _M_initialize_numpunct (__c_locale __cloc)`
- virtual `char_type do_decimal_point () const`
- virtual `string_type do_falsename () const`
- virtual `string do_grouping () const`
- virtual `char_type do_thousands_sep () const`
- virtual `string_type do_truename () const`

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char * __s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

#### Protected Attributes

- `__cache_type * _M_data`

#### 4.912.1 Detailed Description

`template<typename _CharT> class std::numpunct<_CharT>`

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.

The `numpunct` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a `numpunct` facet.

Definition at line 1641 of file `locale_facets.h`.

#### 4.912.2 Member Typedef Documentation

4.912.2.1 `template<typename _CharT> typedef _CharT std::numpunct<_CharT>::char_type`

Public typedefs.

Definition at line 1647 of file `locale_facets.h`.

4.912.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::numpunct<_CharT>::string_type`

Public typedefs.

Definition at line 1648 of file `locale_facets.h`.

## 4.912.3 Constructor &amp; Destructor Documentation

4.912.3.1 `template<typename _CharT> std::numpunct<_CharT>::numpunct ( size_t __refs = 0 ) [inline], [explicit]`

Numpunct constructor.

## Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__refs</code> | RefCount to pass to the base class. |
|---------------------|-------------------------------------|

Definition at line 1665 of file locale\_facets.h.

4.912.3.2 `template<typename _CharT> std::numpunct<_CharT>::numpunct ( __cache_type * __cache, size_t __refs = 0 ) [inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

## Parameters

|                      |                                       |
|----------------------|---------------------------------------|
| <code>__cache</code> | <code>__numpunct_cache</code> object. |
| <code>__refs</code>  | RefCount to pass to the base class.   |

Definition at line 1679 of file locale\_facets.h.

4.912.3.3 `template<typename _CharT> std::numpunct<_CharT>::numpunct ( _c_locale __cloc, size_t __refs = 0 ) [inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__refs</code> | RefCount to pass to the base class. |

Definition at line 1693 of file locale\_facets.h.

4.912.3.4 `template<typename _CharT> virtual std::numpunct<_CharT>::~~numpunct ( ) [protected], [virtual]`

Destructor.

## 4.912.4 Member Function Documentation

4.912.4.1 `template<typename _CharT> char_type std::numpunct<_CharT>::decimal_point ( ) const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `numpunct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1707 of file locale\_facets.h.

**4.912.4.2** `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_decimal_point ( ) const`  
`[inline], [protected], [virtual]`

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1794 of file locale\_facets.h.

**4.912.4.3** `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename ( ) const`  
`[inline], [protected], [virtual]`

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1845 of file locale\_facets.h.

**4.912.4.4** `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping ( ) const` `[inline],`  
`[protected], [virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See Also**

`grouping()` for details.

**Returns**

String representing grouping specification.

Definition at line 1819 of file locale\_facets.h.

**4.912.4.5** `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep ( ) const`  
`[inline], [protected], [virtual]`

Return thousands separator character.

Returns a *char\_type* to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1806 of file locale\_facets.h.

4.912.4.6 `template<typename _CharT> virtual string_type std::numpunct<_CharT>::do_truename ( ) const`  
`[inline], [protected], [virtual]`

Return string representation of bool true.

Returns a *string\_type* containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of true.

Definition at line 1832 of file locale\_facets.h.

4.912.4.7 `template<typename _CharT> string_type std::numpunct<_CharT>::falsename ( ) const` `[inline]`

Return string representation of bool false.

This function returns a *string\_type* containing the text representation for false bool variables. It does so by calling `numpunct<char_type>::do_falsename()`.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1777 of file locale\_facets.h.

4.912.4.8 `template<typename _CharT> string std::numpunct<_CharT>::grouping ( ) const` `[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `numpunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

Definition at line 1751 of file locale\_facets.h.

4.912.4.9 `template<typename _CharT> char_type std::numpunct<_CharT>::thousands_sep ( ) const` `[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `numpunct<char_type>::do_thousands_sep()`.

#### Returns

`char_type` representing a thousands separator.

Definition at line 1720 of file `locale_facets.h`.

4.912.4.10 `template<typename _CharT> string_type std::numpunct<_CharT>::truename( ) const [inline]`

Return string representation of `bool true`.

This function returns a `string_type` containing the text representation for `true` `bool` variables. It does so by calling `numpunct<char_type>::do_truename()`.

#### Returns

`string_type` representing printed form of `true`.

Definition at line 1764 of file `locale_facets.h`.

### 4.912.5 Member Data Documentation

4.912.5.1 `template<typename _CharT> locale::id std::numpunct<_CharT>::id [static]`

Numpunct facet id.

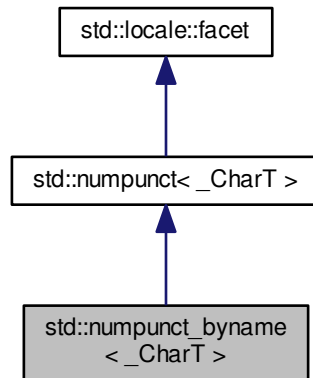
Definition at line 1657 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.913 std::numpunct\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::numpunct\_byname<\_CharT>:



## Public Types

- typedef \_\_numpunct\_cache<\_CharT> \_\_cache\_type
- typedef \_CharT char\_type
- typedef basic\_string<\_CharT> string\_type

## Public Member Functions

- numpunct\_byname (const char \*\_\_s, size\_t \_\_refs=0)
- char\_type decimal\_point () const
- string\_type falsename () const
- string grouping () const
- char\_type thousands\_sep () const
- string\_type truename () const

## Static Public Attributes

- static locale::id id

## Protected Member Functions

- void \_M\_initialize\_numpunct (\_\_c\_locale \_\_cloc=0)
- template<>  
void \_M\_initialize\_numpunct (\_\_c\_locale \_\_cloc)
- template<>  
void \_M\_initialize\_numpunct (\_\_c\_locale \_\_cloc)

- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### Protected Attributes

- `__cache_type * _M_data`

#### 4.913.1 Detailed Description

template<typename \_CharT>class std::num\_punct\_byname< \_CharT >

class num\_punct\_byname [22.2.3.2].

Definition at line 1874 of file locale\_facets.h.

#### 4.913.2 Member Function Documentation

4.913.2.1 template<typename \_CharT> char\_type std::num\_punct< \_CharT >::decimal\_point ( ) const [inline], [inherited]

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning num\_punct<char\_type>::do\_decimal\_point().

##### Returns

*char\_type* representing a decimal point.

Definition at line 1707 of file locale\_facets.h.

4.913.2.2 template<typename \_CharT> virtual char\_type std::num\_punct< \_CharT >::do\_decimal\_point ( ) const [inline], [protected], [virtual], [inherited]

Return decimal point character.

Returns a char\_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1794 of file locale\_facets.h.

4.913.2.3 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename ( ) const`  
[inline], [protected], [virtual], [inherited]

Return string representation of bool false.

Returns a string\_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

#### Returns

string\_type representing printed form of false.

Definition at line 1845 of file locale\_facets.h.

4.913.2.4 `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping ( ) const` [inline],  
[protected], [virtual], [inherited]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

#### See Also

grouping() for details.

#### Returns

String representing grouping specification.

Definition at line 1819 of file locale\_facets.h.

4.913.2.5 `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep ( ) const`  
[inline], [protected], [virtual], [inherited]

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

char\_type representing a thousands separator.

Definition at line 1806 of file locale\_facets.h.

4.913.2.6 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_truename ( ) const`  
[inline], [protected], [virtual], [inherited]

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

#### Returns

string\_type representing printed form of true.

Definition at line 1832 of file locale\_facets.h.

4.913.2.7 `template<typename _CharT> string_type std::num_punct<_CharT>::false_name ( ) const [inline],  
[inherited]`

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_false\_name().

#### Returns

string\_type representing printed form of false.

Definition at line 1777 of file locale\_facets.h.

4.913.2.8 `template<typename _CharT> string std::num_punct<_CharT>::grouping ( ) const [inline],  
[inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

#### Returns

string representing grouping specification.

Definition at line 1751 of file locale\_facets.h.

4.913.2.9 `template<typename _CharT> char_type std::num_punct<_CharT>::thousands_sep ( ) const [inline],  
[inherited]`

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

#### Returns

char\_type representing a thousands separator.

Definition at line 1720 of file locale\_facets.h.

4.913.2.10 `template<typename _CharT> string_type std::num_punct<_CharT>::true_name ( ) const [inline],  
[inherited]`

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling num\_punct<char\_type>::do\_true\_name().

## Returns

string\_type representing printed form of true.

Definition at line 1764 of file locale\_facets.h.

## 4.913.3 Member Data Documentation

4.913.3.1 `template<typename _CharT> locale::id std::numpunct<_CharT>::id` [static], [inherited]

Numpunct facet id.

Definition at line 1657 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.914 std::once\_flag Struct Reference

## Public Member Functions

- constexpr [once\\_flag](#) () noexcept=default
- [once\\_flag](#) (const [once\\_flag](#) &)=delete
- [once\\_flag](#) & operator= (const [once\\_flag](#) &)=delete

## Friends

- `template<typename _Callable, typename... _Args>`  
void [call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&...\_\_args)

## 4.914.1 Detailed Description

[once\\_flag](#)

Definition at line 723 of file mutex.

## 4.914.2 Constructor &amp; Destructor Documentation

4.914.2.1 `constexpr std::once_flag::once_flag ( )` [default], [noexcept]

Constructor.

4.914.2.2 `std::once_flag::once_flag ( const once\_flag & )` [delete]

Deleted copy constructor.

## 4.914.3 Member Function Documentation

4.914.3.1 `once\_flag & std::once_flag::operator= ( const once\_flag & )` [delete]

Deleted assignment operator.

## 4.914.4 Friends And Related Function Documentation

4.914.4.1 `template<typename _Callable, typename... _Args> void call_once ( once_flag & __once, _Callable && __f, _Args &&... __args ) [friend]`

call\_once

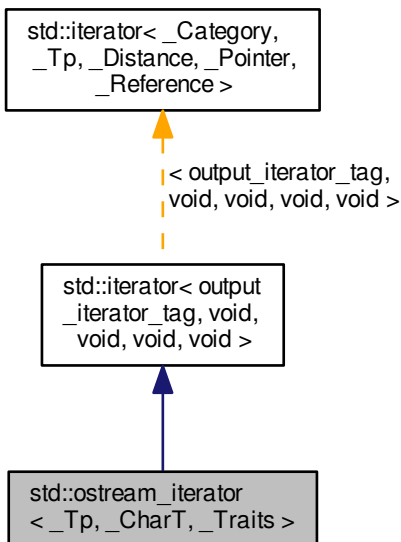
Definition at line 768 of file mutex.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 4.915 std::ostream\_iterator&lt; \_Tp, \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::ostream\_iterator< \_Tp, \_CharT, \_Traits >:



## Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
  
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)

- typedef [basic\\_ostream](#)< \_CharT, \_Traits > [ostream\\_type](#)

#### Public Member Functions

- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s)
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const \_CharT \*\_\_c)
- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj)
- [ostream\\_iterator](#) & **operator\*** ()
- [ostream\\_iterator](#) & **operator++** ()
- [ostream\\_iterator](#) & **operator++** (int)
- [ostream\\_iterator](#) & **operator=** (const \_Tp &\_\_value)

#### 4.915.1 Detailed Description

template<typename \_Tp, typename \_CharT = char, typename \_Traits = char\_traits<\_CharT>>class std::ostream\_iterator< \_Tp, \_CharT, \_Traits >

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type Tp is the only type written by this iterator and there must be an operator<<(Tp) defined.

#### Template Parameters

|                         |                                   |
|-------------------------|-----------------------------------|
| <a href="#">_Tp</a>     | The type to write to the ostream. |
| <a href="#">_CharT</a>  | The ostream char_type.            |
| <a href="#">_Traits</a> | The ostream char_traits.          |

Definition at line 154 of file stream\_iterator.h.

#### 4.915.2 Member Typedef Documentation

4.915.2.1 template<typename \_Tp , typename \_CharT = char, typename \_Traits = char\_traits<\_CharT>> typedef \_CharT std::ostream\_iterator< \_Tp, \_CharT, \_Traits >::char\_type

Public typedef.

Definition at line 160 of file stream\_iterator.h.

4.915.2.2 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference\_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

4.915.2.3 typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void , void , void , void >::iterator\_category [inherited]

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

4.915.2.4 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 162 of file stream\_iterator.h.

4.915.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

4.915.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

4.915.2.7 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 161 of file stream\_iterator.h.

4.915.2.8 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

### 4.915.3 Constructor & Destructor Documentation

4.915.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator( ostream_type & __s ) [inline]`

Construct from an ostream.

Definition at line 171 of file stream\_iterator.h.

4.915.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator( ostream_type & __s, const _CharT* __c ) [inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

#### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__s</code> | Underlying ostream to write to.          |
| <code>__c</code> | <i>CharT</i> delimiter string to insert. |

Definition at line 183 of file stream\_iterator.h.

4.915.3.3 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator<_Tp, _CharT, _Traits>::ostream_iterator ( const ostream_iterator<_Tp, _CharT, _Traits> & __obj ) [inline]`

Copy constructor.

Definition at line 187 of file stream\_iterator.h.

#### 4.915.4 Member Function Documentation

4.915.4.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> ostream_iterator& std::ostream_iterator<_Tp, _CharT, _Traits>::operator= ( const _Tp & __value ) [inline]`

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.

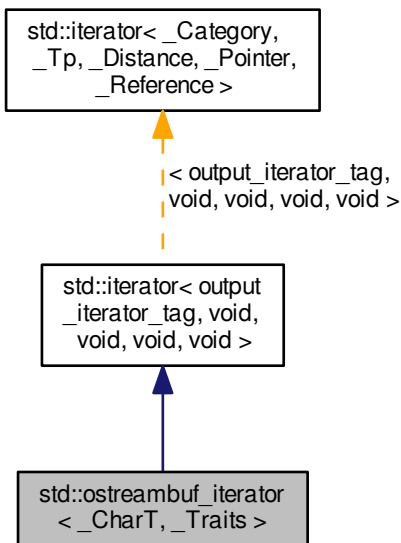
Definition at line 193 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 4.916 std::ostreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::ostreambuf\_iterator< \_CharT, \_Traits >:



#### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)

- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_streambuf](#)  
    < [\\_CharT](#), [\\_Traits](#) > [streambuf\\_type](#)
- typedef [basic\\_ostream](#)< [\\_CharT](#),  
    [\\_Traits](#) > [ostream\\_type](#)

#### Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const [\\_CharT](#) \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const noexcept
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator=](#) ([\\_CharT](#) \_\_c)

#### Friends

- template<typename [\\_CharT2](#) >  
    [\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)  
    < [\\_\\_is\\_char](#)< [\\_CharT2](#) >  
    ::value, [ostreambuf\\_iterator](#)  
    < [\\_CharT2](#) > >::type **copy** ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >)

#### 4.916.1 Detailed Description

template<typename [\\_CharT](#), typename [\\_Traits](#)>class std::ostreambuf\_iterator< [\\_CharT](#), [\\_Traits](#) >

Provides output iterator semantics for streambufs.

Definition at line 216 of file streambuf\_iterator.h.

#### 4.916.2 Member Typedef Documentation

4.916.2.1 template<typename [\\_CharT](#) , typename [\\_Traits](#) > typedef [\\_CharT](#) std::ostreambuf\_iterator< [\\_CharT](#), [\\_Traits](#) >::char\_type

Public typedefs.

Definition at line 223 of file streambuf\_iterator.h.

4.916.2.2 typedef void std::iterator< [output\\_iterator\\_tag](#) , void , void , void , void >::difference\_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

4.916.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.916.2.4 `template<typename _CharT, typename _Traits > typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type`

Public typedefs.

Definition at line 226 of file `streambuf_iterator.h`.

4.916.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.916.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference` [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.916.2.7 `template<typename _CharT, typename _Traits > typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 225 of file `streambuf_iterator.h`.

4.916.2.8 `template<typename _CharT, typename _Traits > typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 224 of file `streambuf_iterator.h`.

4.916.2.9 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

#### 4.916.3 Constructor & Destructor Documentation

4.916.3.1 `template<typename _CharT, typename _Traits > std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator( ostream_type & __s )` [inline], [noexcept]

Construct output iterator from ostream.

Definition at line 241 of file `streambuf_iterator.h`.

4.916.3.2 `template<typename _CharT, typename _Traits > std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator( streambuf_type * __s )` [inline], [noexcept]

Construct output iterator from streambuf.

Definition at line 245 of file ostreambuf\_iterator.h.

#### 4.916.4 Member Function Documentation

4.916.4.1 `template<typename _CharT, typename _Traits> bool std::ostreambuf_iterator<_CharT, _Traits>::failed( ) const [inline], [noexcept]`

Return true if previous operator=() failed.

Definition at line 275 of file ostreambuf\_iterator.h.

4.916.4.2 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator*( ) [inline]`

Return \*this.

Definition at line 260 of file ostreambuf\_iterator.h.

4.916.4.3 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator++( int ) [inline]`

Return \*this.

Definition at line 265 of file ostreambuf\_iterator.h.

4.916.4.4 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator++( ) [inline]`

Return \*this.

Definition at line 270 of file ostreambuf\_iterator.h.

4.916.4.5 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator=( _CharT __c ) [inline]`

Write character to ostreambuf. Calls ostreambuf.sputc().

Definition at line 250 of file ostreambuf\_iterator.h.

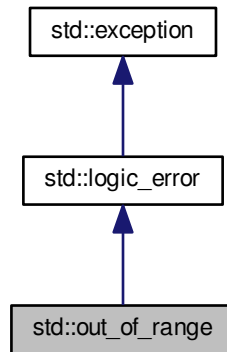
References std::basic\_ostreambuf<\_CharT, \_Traits>::sputc().

The documentation for this class was generated from the following file:

- [ostreambuf\\_iterator.h](#)

## 4.917 `std::out_of_range` Class Reference

Inheritance diagram for `std::out_of_range`:



### Public Member Functions

- **`out_of_range`** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

### 4.917.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in `basic_string`).

Definition at line 100 of file `stdexcept`.

### 4.917.2 Member Function Documentation

#### 4.917.2.1 virtual const char\* `std::logic_error::what` ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 4.918 `std::output_iterator_tag` Struct Reference

### 4.918.1 Detailed Description

Marking output iterators.

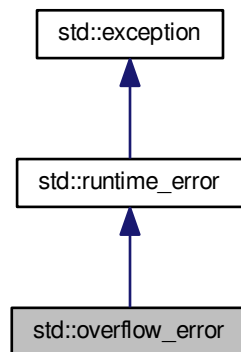
Definition at line 92 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.919 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- **`overflow_error`** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

#### 4.919.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 138 of file `stdexcept`.

#### 4.919.2 Member Function Documentation

**4.919.2.1** virtual const char\* `std::runtime_error::what ( ) const` [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 4.920 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::_Sp_owner_less< _Tp, _Tp1 >`.

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool operator()` (`const _Tp &__lhs, const _Tp &__rhs`) `const`
- `bool operator()` (`const _Tp &__lhs, const _Tp1 &__rhs`) `const`
- `bool operator()` (`const _Tp1 &__lhs, const _Tp &__rhs`) `const`

#### 4.920.1 Detailed Description

```
template<typename _Tp>struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for `shared_ptr`.

Definition at line 527 of file `shared_ptr.h`.

#### 4.920.2 Member Typedef Documentation

4.920.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.920.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.920.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 4.921 `std::owner_less< weak_ptr< _Tp > >` Struct Template Reference

Inherits `std::_Sp_owner_less< _Tp, _Tp1 >`.

## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool operator()` (`const _Tp &__lhs, const _Tp &__rhs`) `const`
- `bool operator()` (`const _Tp &__lhs, const _Tp1 &__rhs`) `const`
- `bool operator()` (`const _Tp1 &__lhs, const _Tp &__rhs`) `const`

### 4.921.1 Detailed Description

```
template<typename _Tp>struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 533 of file `shared_ptr.h`.

### 4.921.2 Member Typedef Documentation

4.921.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.921.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.921.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

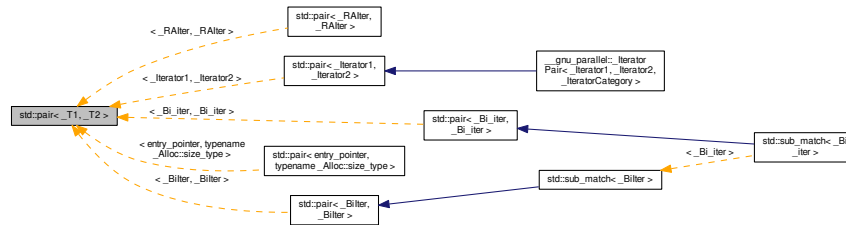
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 4.922 std::pair&lt; \_T1, \_T2 &gt; Struct Template Reference

Inheritance diagram for std::pair< \_T1, \_T2 >:



## Public Types

- typedef **\_T1** **first\_type**
- typedef **\_T2** **second\_type**

## Public Member Functions

- constexpr **pair** ()
- constexpr **pair** (const **\_T1** &\_\_a, const **\_T2** &\_\_b)
- template<class **\_U1** , class **\_U2** , class = typename enable\_if<\_\_and<is\_convertible<const **\_U1**&, **\_T1**>, is\_convertible<const **\_U2**&, **\_T2**>>::value>::type>  
constexpr **pair** (const **pair**< **\_U1**, **\_U2** > &\_\_p)
- constexpr **pair** (const **pair** &)=default
- constexpr **pair** (**pair** &&)=default
- template<class **\_U1** , class = typename enable\_if<is\_convertible< **\_U1**, **\_T1**>::value>::type>  
constexpr **pair** ( **\_U1** &&\_\_x, const **\_T2** &\_\_y)
- template<class **\_U2** , class = typename enable\_if<is\_convertible< **\_U2**, **\_T2**>::value>::type>  
constexpr **pair** (const **\_T1** &\_\_x, **\_U2** &&\_\_y)
- template<class **\_U1** , class **\_U2** , class = typename enable\_if<\_\_and<is\_convertible< **\_U1**, **\_T1**>, is\_convertible< **\_U2**, **\_T2**>>::value>::type>  
constexpr **pair** ( **\_U1** &&\_\_x, **\_U2** &&\_\_y)
- template<class **\_U1** , class **\_U2** , class = typename enable\_if<\_\_and<is\_convertible< **\_U1**, **\_T1**>, is\_convertible< **\_U2**, **\_T2**>>::value>::type>  
constexpr **pair** (**pair**< **\_U1**, **\_U2** > &&\_\_p)
- template<typename... **\_Args1**, typename... **\_Args2**>  
**pair** (**piecewise\_construct\_t**, **tuple**< **\_Args1**...>, **tuple**< **\_Args2**...>)
- void \_\_p **first** && **noexcept** (swap(**second**, \_\_p.second)))
- **pair** & **operator=** (const **pair** &\_\_p)
- **pair** & **operator=** (**pair** &&\_\_p) noexcept(\_\_and< **is\_nothrow\_move\_assignable**< **\_T1** >
- template<class **\_U1** , class **\_U2** >  
**pair** & **operator=** (const **pair**< **\_U1**, **\_U2** > &\_\_p)
- template<class **\_U1** , class **\_U2** >  
**pair** & **operator=** (**pair**< **\_U1**, **\_U2** > &&\_\_p)
- void **swap** (**pair** &\_\_p) noexcept(noexcept(swap(**first**

## Public Attributes

- `_T1` [first](#)
- [pair](#)  
[is\\_nothrow\\_move\\_assignable](#)  
`<_T2>::value` **first**
- `_T2` [second](#)
- **second**
- `return` \* **this**

## 4.922.1 Detailed Description

```
template<class _T1, class _T2>struct std::pair< _T1, _T2 >
```

Struct holding two objects of arbitrary type.

## Template Parameters

|                  |                        |
|------------------|------------------------|
| <code>_T1</code> | Type of first object.  |
| <code>_T2</code> | Type of second object. |

Definition at line 96 of file `stl_pair.h`.

## 4.922.2 Member Typedef Documentation

4.922.2.1 `template<class _T1, class _T2> typedef _T2 std::pair< _T1, _T2 >::second_type`

`first_type` is the first bound type

Definition at line 99 of file `stl_pair.h`.

## 4.922.3 Constructor &amp; Destructor Documentation

4.922.3.1 `template<class _T1, class _T2> constexpr std::pair< _T1, _T2 >::pair ( ) [inline]`

`second` is a copy of the second object

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 108 of file `stl_pair.h`.

4.922.3.2 `template<class _T1, class _T2> constexpr std::pair< _T1, _T2 >::pair ( const _T1 & __a, const _T2 & __b ) [inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 112 of file `stl_pair.h`.

4.922.3.3 `template<class _T1, class _T2> template<class _U1 , class _U2 , class = typename enable_if<__and_<is_convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type> constexpr std::pair< _T1, _T2 >::pair ( const pair< _U1, _U2 > & __p ) [inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 124 of file `stl_pair.h`.

## 4.922.4 Member Data Documentation

4.922.4.1 `template<class _T1, class _T2> _T1 std::pair<_T1, _T2>::first`

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, `std::operator<()`, and `std::operator==()`.

4.922.4.2 `template<class _T1, class _T2> _T2 std::pair<_T1, _T2>::second`

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, and `std::operator==()`.

The documentation for this struct was generated from the following files:

- [stl\\_pair.h](#)
- [tuple](#)

4.923 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**`piecewise_constant_distribution`** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**`piecewise_constant_distribution`** (`initializer_list<_RealType> __bl, _Func __fw`)
- `template<typename _Func >`  
**`piecewise_constant_distribution`** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **`piecewise_constant_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
**`void __generate`** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
**`void __generate`** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
**`void __generate`** (`result\_type * __f, result\_type * __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `std::vector<double> densities () const`
- `std::vector<_RealType> intervals () const`

- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >  
std::basic_ostream<_CharT,  
_Traits > & operator<< (std::basic_ostream<_CharT, _Traits > &__os, const std::piecewise_constant_  
distribution<_RealType1 > &__x)`
- `bool operator== (const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >  
std::basic_istream<_CharT,  
_Traits > & operator>> (std::basic_istream<_CharT, _Traits > &__is, std::piecewise_constant_distribution<  
_RealType1 > &__x)`

#### 4.923.1 Detailed Description

`template<typename _RealType = double>class std::piecewise_constant_distribution<_RealType>`

A `piecewise_constant_distribution` random number distribution.

The formula for the piecewise constant probability mass function is

Definition at line 5480 of file `random.h`.

#### 4.923.2 Member Typedef Documentation

4.923.2.1 `template<typename _RealType = double> typedef _RealType std::piecewise_constant_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 5483 of file `random.h`.

#### 4.923.3 Member Function Documentation

4.923.3.1 `template<typename _RealType = double> std::vector<double> std::piecewise_constant_distribution<_RealType>::densities ( ) const [inline]`

Returns a vector of the probability densities.

Definition at line 5601 of file `random.h`.

References `std::vector<_Tp, _Alloc>::empty()`.

4.923.3.2 `template<typename _RealType = double> std::vector<_RealType> std::piecewise_constant_distribution<_RealType>::intervals ( ) const [inline]`

Returns a vector of the intervals.

Definition at line 5585 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

4.923.3.3 `template<typename _RealType = double> result_type std::piecewise_constant_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5636 of file random.h.

References `std::vector<_Tp, _Alloc>::back()`, and `std::vector<_Tp, _Alloc>::empty()`.

4.923.3.4 `template<typename _RealType = double> result_type std::piecewise_constant_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5626 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`, and `std::vector<_Tp, _Alloc>::front()`.

4.923.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::piecewise_constant_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5647 of file random.h.

References `std::piecewise_constant_distribution<_RealType>::operator()()`.

Referenced by `std::piecewise_constant_distribution<_RealType>::operator()()`.

4.923.3.6 `template<typename _RealType = double> param_type std::piecewise_constant_distribution<_RealType>::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 5611 of file random.h.

4.923.3.7 `template<typename _RealType = double> void std::piecewise_constant_distribution<_RealType>::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 5619 of file random.h.

4.923.3.8 `template<typename _RealType = double> void std::piecewise_constant_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 5578 of file `random.h`.

#### 4.923.4 Friends And Related Function Documentation

4.923.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::piecewise_constant_distribution<_RealType1> & __x ) [friend]`

Inserts a `piecewise_constant_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                          |
| <code>__x</code>  | A <code>piecewise_constant_distribution</code> random number distribution. |

##### Returns

The output stream with the state of `__x` inserted or in an error state.

4.923.4.2 `template<typename _RealType = double> bool operator==( const piecewise_constant_distribution<_RealType> & __d1, const piecewise_constant_distribution<_RealType> & __d2 ) [friend]`

Return true if two `piecewise_constant_distribution` have the same parameters.

Definition at line 5682 of file `random.h`.

4.923.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::piecewise_constant_distribution<_RealType1> & __x ) [friend]`

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

|                   |                                                                                |
|-------------------|--------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                               |
| <code>__x</code>  | A <code>piecewise_constant_distribution</code> random number generator engine. |

##### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.924 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef  
[piecewise\\_constant\\_distribution](#)  
<\_RealType> **distribution\_type**

## Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**param\_type** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**param\_type** (`initializer_list<_RealType> __bi, _Func __fw`)
- `template<typename _Func >`  
**param\_type** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **param\_type** (`const param_type &`)=default
- `std::vector<double>` **densities** () const
- `std::vector<_RealType>` **intervals** () const
- `param_type &` **operator=** (`const param_type &`)=default

## Friends

- `bool` **operator==** (`const param_type &__p1, const param_type &__p2`)
- `class` **piecewise\_constant\_distribution<\_RealType>**

## 4.924.1 Detailed Description

`template<typename _RealType = double>struct std::piecewise_constant_distribution<_RealType>::param_type`

Parameter type.

Definition at line 5489 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.925 std::piecewise\_construct\_t Struct Reference

## 4.925.1 Detailed Description

`piecewise_construct_t`

Definition at line 76 of file `stl_pair.h`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 4.926 std::piecewise\_linear\_distribution&lt;\_RealType&gt; Class Template Reference

## Classes

- `struct` [param\\_type](#)

## Public Types

- `typedef _RealType` [result\\_type](#)

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**piecewise\_linear\_distribution** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**piecewise\_linear\_distribution** (initializer\_list<\_RealType> \_\_bl, \_Func \_\_fw)
- template<typename \_Func >  
**piecewise\_linear\_distribution** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **piecewise\_linear\_distribution** (const param\_type &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** (result\_type \*\_\_f, result\_type \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- std::vector<double> **densities** () const
- std::vector<\_RealType> **intervals** () const
- result\_type **max** () const
- result\_type **min** () const
- template<typename \_UniformRandomNumberGenerator >  
result\_type **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
result\_type **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- param\_type **param** () const
- void **param** (const param\_type &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
std::basic\_ostream<\_CharT, \_Traits> & **operator<<** (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const std::piecewise\_linear\_distribution<\_RealType1> &\_\_x)
- bool **operator==** (const piecewise\_linear\_distribution &\_\_d1, const piecewise\_linear\_distribution &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
std::basic\_istream<\_CharT, \_Traits> & **operator>>** (std::basic\_istream<\_CharT, \_Traits> &\_\_is, std::piecewise\_linear\_distribution<\_RealType1> &\_\_x)

## 4.926.1 Detailed Description

template<typename \_RealType = double>class std::piecewise\_linear\_distribution<\_RealType>

A piecewise\_linear\_distribution random number distribution.

The formula for the piecewise linear probability mass function is

Definition at line 5747 of file random.h.

#### 4.926.2 Member Typedef Documentation

4.926.2.1 `template<typename _RealType = double> typedef _RealType std::piecewise_linear_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 5750 of file random.h.

#### 4.926.3 Member Function Documentation

4.926.3.1 `template<typename _RealType = double> std::vector<double> std::piecewise_linear_distribution<_RealType>::densities ( ) const [inline]`

Return a vector of the probability densities of the distribution.

Definition at line 5871 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

4.926.3.2 `template<typename _RealType = double> std::vector<_RealType> std::piecewise_linear_distribution<_RealType>::intervals ( ) const [inline]`

Return the intervals of the distribution.

Definition at line 5854 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

4.926.3.3 `template<typename _RealType = double> result_type std::piecewise_linear_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5906 of file random.h.

References `std::vector<_Tp, _Alloc>::back()`, and `std::vector<_Tp, _Alloc>::empty()`.

4.926.3.4 `template<typename _RealType = double> result_type std::piecewise_linear_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5896 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`, and `std::vector<_Tp, _Alloc>::front()`.

4.926.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::piecewise_linear_distribution<_RealType>::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5917 of file random.h.

References `std::piecewise_linear_distribution<_RealType>::operator()()`.

Referenced by `std::piecewise_linear_distribution<_RealType>::operator()()`.

4.926.3.6 `template<typename _RealType = double> param_type std::piecewise_linear_distribution<_RealType>::param ( ) const` `[inline]`

Returns the parameter set of the distribution.

Definition at line 5881 of file random.h.

4.926.3.7 `template<typename _RealType = double> void std::piecewise_linear_distribution<_RealType>::param ( const param_type & __param )` `[inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 5889 of file random.h.

4.926.3.8 `template<typename _RealType = double> void std::piecewise_linear_distribution<_RealType>::reset ( )` `[inline]`

Resets the distribution state.

Definition at line 5847 of file random.h.

#### 4.926.4 Friends And Related Function Documentation

4.926.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::piecewise_linear_distribution<_RealType1> & __x )` `[friend]`

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                          |
|-------------------|--------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                        |
| <code>__x</code>  | A <code>piecewise_linear_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.926.4.2 `template<typename _RealType = double> bool operator==( const piecewise_linear_distribution<_RealType> & __d1, const piecewise_linear_distribution<_RealType> & __d2 )` `[friend]`

Return true if two `piecewise_linear_distribution` have the same parameters.

Definition at line 5952 of file random.h.

4.926.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::piecewise_linear_distribution<_RealType1> & __x )` `[friend]`

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                             |
| <code>__x</code>  | A <code>piecewise_linear_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.927 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef  
[piecewise\\_linear\\_distribution](#)  
<\_RealType> **distribution\_type**

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**param\_type** ([initializer\\_list](#)<\_RealType> \_\_bl, \_Func \_\_fw)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [std::vector](#)<double> **densities** () const
- [std::vector](#)<\_RealType> **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **piecewise\_linear\_distribution**<\_RealType>

## 4.927.1 Detailed Description

```
template<typename _RealType = double>struct std::piecewise_linear_distribution<_RealType>::param_type
```

Parameter type.

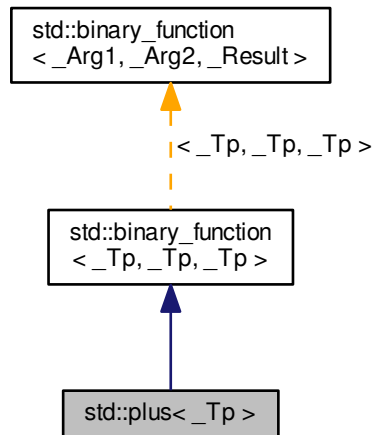
Definition at line 5756 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.928 std::plus&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::plus< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

## 4.928.1 Detailed Description

```
template<typename _Tp>struct std::plus< _Tp >
```

One of the [math functors](#).

Definition at line 140 of file `stl_function.h`.

## 4.928.2 Member Typedef Documentation

4.928.2.1 typedef `_Tp` `std::binary_function<_Tp, _Tp, _Tp>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.928.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` [inherited]

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.928.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

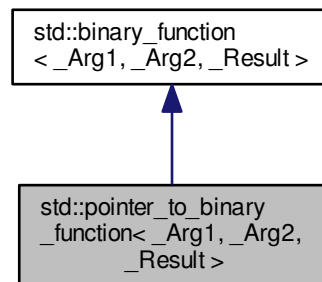
Definition at line 120 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.929 `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >` Class Template Reference

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



### Public Types

- `typedef _Arg1` [first\\_argument\\_type](#)
- `typedef _Result` [result\\_type](#)
- `typedef _Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- **`pointer_to_binary_function`** (`_Result(*__x)(_Arg1, _Arg2)`)
- `_Result` **`operator()`** (`_Arg1 __x, _Arg2 __y`) `const`

### Protected Attributes

- `_Result(* _M_ptr)(_Arg1, _Arg2)`

## 4.929.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 447 of file `stl_function.h`.

## 4.929.2 Member Typedef Documentation

4.929.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 117 of file `stl_function.h`.

4.929.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 123 of file `stl_function.h`.

4.929.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

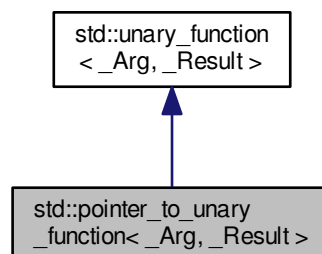
Definition at line 120 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

4.930 `std::pointer_to_unary_function< _Arg, _Result >` Class Template Reference

Inheritance diagram for `std::pointer_to_unary_function< _Arg, _Result >`:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## Public Member Functions

- **`pointer_to_unary_function`** (`_Result(*__x)(_Arg)`)
- `_Result` **`operator()`** (`_Arg __x`) const

## Protected Attributes

- `_Result(*_M_ptr)(_Arg)`

## 4.930.1 Detailed Description

```
template<typename _Arg, typename _Result> class std::pointer_to_unary_function< _Arg, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 422 of file `stl_function.h`.

## 4.930.2 Member Typedef Documentation

4.930.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`  
[[inherited](#)]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.930.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`  
[[inherited](#)]

`result_type` is the return type

Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

4.931 `std::pointer_traits<_Ptr>` Struct Template Reference

Inherits `std::__ptrtr_pointer_to<_Ptr>`.

## Public Types

- typedef `__ptrtr_diff_type`  
`<_Ptr>::__type` [difference\\_type](#)
- typedef `__ptrtr_elt_type<_Ptr>`  
`::__type` [element\\_type](#)

- typedef `_Ptr` [pointer](#)
- template<typename `_Up`>  
using **rebind** = typename `__ptrtr_rebind<_Ptr, _Up>::__type`

#### Static Public Member Functions

- static `_Ptr` **pointer\_to** (`__element_type` &`__e`)

#### 4.931.1 Detailed Description

template<typename `_Ptr`>struct `std::pointer_traits<_Ptr>`

Uniform interface to all pointer-like types.

Definition at line 137 of file `ptr_traits.h`.

#### 4.931.2 Member Typedef Documentation

4.931.2.1 template<typename `_Ptr`> typedef `__ptrtr_diff_type<_Ptr>::__type` `std::pointer_traits<_Ptr>::difference_type`

Type used to represent the difference between two pointers.

Definition at line 144 of file `ptr_traits.h`.

4.931.2.2 template<typename `_Ptr`> typedef `__ptrtr_elt_type<_Ptr>::__type` `std::pointer_traits<_Ptr>::element_type`

The type pointed to.

Definition at line 142 of file `ptr_traits.h`.

4.931.2.3 template<typename `_Ptr`> typedef `_Ptr` `std::pointer_traits<_Ptr>::pointer`

The pointer type.

Definition at line 140 of file `ptr_traits.h`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

## 4.932 `std::pointer_traits<_Tp*>` Struct Template Reference

### Public Types

- typedef `ptrdiff_t` [difference\\_type](#)
- typedef `_Tp` [element\\_type](#)
- typedef `_Tp*` [pointer](#)
- template<typename `_Up`>  
using **rebind** = `_Up*`

### Static Public Member Functions

- static [pointer](#) **pointer\_to** (typename `__ptrtr_not_void<element_type>::__type` &`__r`) noexcept

## 4.932.1 Detailed Description

```
template<typename _Tp>struct std::pointer_traits<_Tp*>
```

Partial specialization for built-in pointers.

Definition at line 155 of file `ptr_traits.h`.

## 4.932.2 Member Typedef Documentation

4.932.2.1 `template<typename _Tp> typedef ptrdiff_t std::pointer_traits<_Tp*>::difference_type`

Type used to represent the difference between two pointers.

Definition at line 162 of file `ptr_traits.h`.

4.932.2.2 `template<typename _Tp> typedef _Tp std::pointer_traits<_Tp*>::element_type`

The type pointed to.

Definition at line 160 of file `ptr_traits.h`.

4.932.2.3 `template<typename _Tp> typedef _Tp* std::pointer_traits<_Tp*>::pointer`

The pointer type.

Definition at line 158 of file `ptr_traits.h`.

## 4.932.3 Member Function Documentation

4.932.3.1 `template<typename _Tp> static pointer std::pointer_traits<_Tp*>::pointer_to ( typename __ptrtr_not_void<element_type>::__type & __r ) [inline], [static], [noexcept]`

Obtain a pointer to an object.

## Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__r</code> | A reference to an object of type <code>element_type</code> |
|------------------|------------------------------------------------------------|

## Returns

`addressof(__r)`

Definition at line 173 of file `ptr_traits.h`.

References `std::addressof()`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

4.933 `std::poisson_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- **poisson\_distribution** (double \_\_mean=1.0)
- **poisson\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- double **mean** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::poisson\\_distribution](#)< \_IntType1 > &\_\_x)
- bool **operator==** (const [poisson\\_distribution](#) &\_\_d1, const [poisson\\_distribution](#) &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::poisson\\_distribution](#)< \_IntType1 > &\_\_x)

## 4.933.1 Detailed Description

template<typename \_IntType = int>class [std::poisson\\_distribution](#)< \_IntType >

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Definition at line 4429 of file random.h.

## 4.933.2 Member Typedef Documentation

4.933.2.1 `template<typename _IntType = int> typedef _IntType std::poisson_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4432 of file random.h.

## 4.933.3 Member Function Documentation

4.933.3.1 `template<typename _IntType = int> result_type std::poisson_distribution< _IntType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4523 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

4.933.3.2 `template<typename _IntType = int> double std::poisson_distribution< _IntType >::mean ( ) const [inline]`

Returns the distribution parameter `mean`.

Definition at line 4494 of file random.h.

4.933.3.3 `template<typename _IntType = int> result_type std::poisson_distribution< _IntType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4516 of file random.h.

4.933.3.4 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type std::poisson_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & _urng ) [inline]`

Generating functions.

Definition at line 4531 of file random.h.

References `std::poisson_distribution< _IntType >::operator()()`.

Referenced by `std::poisson_distribution< _IntType >::operator()()`.

4.933.3.5 `template<typename _IntType > template<typename _UniformRandomNumberGenerator > poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & _urng, const param_type & _param )`

A rejection algorithm when `mean >= 12` and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1436 of file bits/random.tcc.

References `std::abs()`, `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::numeric_limits< _Tp >::max()`.

4.933.3.6 `template<typename _IntType = int> param_type std::poisson_distribution< _IntType >::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 4501 of file random.h.

4.933.3.7 `template<typename _IntType = int> void std::poisson_distribution< _IntType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4509 of file random.h.

4.933.3.8 `template<typename _IntType = int> void std::poisson_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4487 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

#### 4.933.4 Friends And Related Function Documentation

4.933.4.1 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::poisson_distribution< _IntType1 > & __x ) [friend]`

Inserts a `poisson_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <code>__os</code> | An output stream.                                               |
| <code>__x</code>  | A <code>poisson_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.933.4.2 `template<typename _IntType = int> bool operator== ( const poisson_distribution< _IntType > & __d1, const poisson_distribution< _IntType > & __d2 ) [friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4567 of file random.h.

4.933.4.3 `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::poisson_distribution< _IntType1 > & __x ) [friend]`

Extracts a `poisson_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                    |
| <code>__x</code>  | A <code>poisson_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.934 `std::poisson_distribution< _IntType >::param_type` Struct Reference

## Public Types

- typedef [poisson\\_distribution](#)  
`< _IntType >` **distribution\_type**

## Public Member Functions

- **param\_type** (double `__mean`=1.0)
- double **mean** () const

## Friends

- bool **operator==** (const [param\\_type](#) &`__p1`, const [param\\_type](#) &`__p2`)
- class **poisson\_distribution**`< _IntType >`

## 4.934.1 Detailed Description

```
template<typename _IntType = int>struct std::poisson_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4438 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.935 `std::priority_queue< _Tp, _Sequence, _Compare >` Class Template Reference

## Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

## Public Member Functions

- `priority_queue` (const `_Compare` &\_\_x, const `_Sequence` &\_\_s)
- `priority_queue` (const `_Compare` &\_\_x=\_Compare(), `_Sequence` &&\_\_s=\_Sequence())
- `template<typename _InputIterator>`  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, const `_Sequence` &\_\_s)
- `template<typename _InputIterator>`  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x=\_Compare(), `_Sequence` &&\_\_s=\_Sequence())
- `template<typename... _Args>`  
void **emplace** (`_Args` &&... \_\_args)
- bool `empty` () const
- void \_\_pq c && **noexcept** (swap(comp, \_\_pq.comp)))
- void `pop` ()
- void `push` (const `value_type` &\_\_x)
- void **push** (`value_type` &&\_\_x)
- `size_type` `size` () const
- void **swap** (`priority_queue` &\_\_pq) noexcept(noexcept(swap(c
- const\_reference `top` () const

## Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

## 4.935.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue<_Tp, _Sequence, _Compare>
```

A standard container automatically sorting its contents.

## Template Parameters

|                        |                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                                              |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>vector&lt;_Tp&gt;</code> .                     |
| <code>_Compare</code>  | Comparison function object type, defaults to <code>less&lt;_Sequence::value_type&gt;</code> . |

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard queue operations.

## Note

No equality/comparison operators are provided for priority\_queue.

Sorting of the elements takes place as they are added to, and removed from, the priority\_queue using the priority\_queue's member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the priority\_queue will not re-sort the elements for you. (How could it know to do so?)

Definition at line 367 of file stl\_queue.h.

## 4.935.2 Constructor &amp; Destructor Documentation

4.935.2.1 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>> std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue ( const _Compare & __x, const _Sequence & __s ) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 402 of file stl\_queue.h.

References std::make\_heap().

4.935.2.2 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>> template<typename _InputIterator> std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue ( _InputIterator __first, _InputIterator __last, const _Compare & __x, const _Sequence & __s ) [inline]`

Builds a queue from a range.

## Parameters

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| <code>__first</code> | An input iterator.                                      |
| <code>__last</code>  | An input iterator.                                      |
| <code>__x</code>     | A comparison functor describing a strict weak ordering. |
| <code>__s</code>     | An initial sequence with which to start.                |

Begins by copying \_\_s, inserting a copy of the elements from [first,last) into the copy of \_\_s, then ordering the copy according to \_\_x.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 442 of file stl\_queue.h.

References std::make\_heap().

## 4.935.3 Member Function Documentation

4.935.3.1 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>> bool std::priority_queue<_Tp, _Sequence, _Compare>::empty ( ) const [inline]`

Returns true if the queue is empty.

Definition at line 468 of file stl\_queue.h.

Referenced by \_\_gnu\_parallel::multiseq\_partition(), and \_\_gnu\_parallel::multiseq\_selection().

4.935.3.2 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> void std::priority_queue< _Tp, _Sequence, _Compare >::pop ( ) [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 531 of file `stl_queue.h`.

References `std::pop_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

4.935.3.3 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> void std::priority_queue< _Tp, _Sequence, _Compare >::push ( const value_type & __x ) [inline]`

Add data to the queue.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 496 of file `stl_queue.h`.

References `std::push_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

4.935.3.4 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> size_type std::priority_queue< _Tp, _Sequence, _Compare >::size ( ) const [inline]`

Returns the number of elements in the queue.

Definition at line 473 of file `stl_queue.h`.

4.935.3.5 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> const_reference std::priority_queue< _Tp, _Sequence, _Compare >::top ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 481 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 4.936 std::queue< \_Tp, \_Sequence > Class Template Reference

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**

- typedef \_Sequence **container\_type**
- typedef \_Sequence::reference **reference**
- typedef \_Sequence::size\_type **size\_type**
- typedef \_Sequence::value\_type **value\_type**

#### Public Member Functions

- [queue](#) (const \_Sequence &\_\_c)
- **queue** (\_Sequence &&\_\_c=\_Sequence())
- reference [back](#) ()
- const\_reference [back](#) () const
- template<typename... \_Args>  
void **emplace** (\_Args &&... \_\_args)
- bool [empty](#) () const
- reference [front](#) ()
- const\_reference [front](#) () const
- void [pop](#) ()
- void [push](#) (const value\_type &\_\_x)
- void **push** (value\_type &&\_\_x)
- size\_type [size](#) () const
- void **swap** ([queue](#) &\_\_q) noexcept(noexcept(swap(c
- **swap** (c, \_\_q.c)

#### Public Attributes

- void \_\_q c

#### Protected Attributes

- \_Sequence c

#### Friends

- template<typename \_Tp1, typename \_Seq1 >  
bool **operator**< (const [queue](#)< \_Tp1, \_Seq1 > &, const [queue](#)< \_Tp1, \_Seq1 > &)
- template<typename \_Tp1, typename \_Seq1 >  
bool **operator**== (const [queue](#)< \_Tp1, \_Seq1 > &, const [queue](#)< \_Tp1, \_Seq1 > &)

#### 4.936.1 Detailed Description

template<typename \_Tp, typename \_Sequence = deque<\_Tp>>class std::queue< \_Tp, \_Sequence >

A standard container giving FIFO behavior.

#### Template Parameters

|                        |                                                      |
|------------------------|------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                     |
| <code>_Sequence</code> | Type of underlying sequence, defaults to deque<_Tp>. |

Meets many of the requirements of a *container*, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 93 of file `stl_queue.h`.

#### 4.936.2 Constructor & Destructor Documentation

4.936.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> std::queue< _Tp, _Sequence >::queue ( const _Sequence &__c ) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 138 of file `stl_queue.h`.

#### 4.936.3 Member Function Documentation

4.936.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::queue< _Tp, _Sequence >::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the queue.

Definition at line 185 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

4.936.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue< _Tp, _Sequence >::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 196 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

4.936.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::queue< _Tp, _Sequence >::empty ( ) const [inline]`

Returns true if the queue is empty.

Definition at line 150 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

4.936.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::queue< _Tp, _Sequence >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 163 of file `stl_queue.h`.

References std::queue< \_Tp, \_Sequence >::c.

**4.936.3.5** `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue< _Tp, _Sequence >::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 174 of file stl\_queue.h.

References std::queue< \_Tp, \_Sequence >::c.

**4.936.3.6** `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue< _Tp, _Sequence >::pop ( ) [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 238 of file stl\_queue.h.

References std::queue< \_Tp, \_Sequence >::c.

**4.936.3.7** `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue< _Tp, _Sequence >::push ( const value_type & __x ) [inline]`

Add data to the end of the queue.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 212 of file stl\_queue.h.

References std::queue< \_Tp, \_Sequence >::c.

**4.936.3.8** `template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::queue< _Tp, _Sequence >::size ( ) const [inline]`

Returns the number of elements in the queue.

Definition at line 155 of file stl\_queue.h.

References std::queue< \_Tp, \_Sequence >::c.

#### 4.936.4 Member Data Documentation

**4.936.4.1** `template<typename _Tp, typename _Sequence = deque<_Tp>> _Sequence std::queue< _Tp, _Sequence >::c [protected]`

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 126 of file stl\_queue.h.

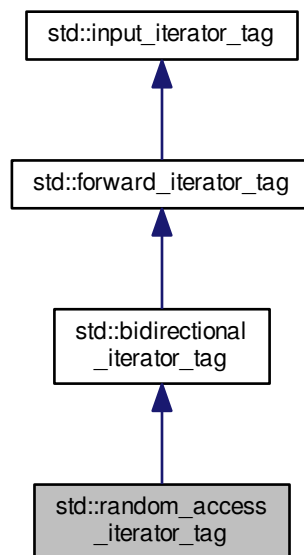
Referenced by `std::queue< _Tp, _Sequence >::back()`, `std::queue< _Tp, _Sequence >::empty()`, `std::queue< _Tp, _Sequence >::front()`, `std::operator<()`, `std::operator==()`, `std::queue< _Tp, _Sequence >::pop()`, `std::queue< _Tp, _Sequence >::push()`, and `std::queue< _Tp, _Sequence >::size()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 4.937 std::random\_access\_iterator\_tag Struct Reference

Inheritance diagram for `std::random_access_iterator_tag`:



### 4.937.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations.

Definition at line 103 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.938 std::random\_device Class Reference

Public Types

- typedef unsigned int [result\\_type](#)

## Public Member Functions

- **random\_device** (const [std::string](#) &\_\_token="mt19937")
- **random\_device** (const [random\\_device](#) &)=delete
- double **entropy** () const noexcept
- [result\\_type](#) **operator()** ()
- void **operator=** (const [random\\_device](#) &)=delete

## Static Public Member Functions

- static constexpr [result\\_type](#) **max** ()
- static constexpr [result\\_type](#) **min** ()

## 4.938.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1575 of file `random.h`.

## 4.938.2 Member Typedef Documentation

4.938.2.1 `typedef unsigned int std::random_device::result_type`

The type of the generated random value.

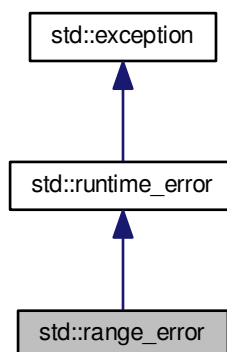
Definition at line 1579 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

4.939 `std::range_error` Class Reference

Inheritance diagram for `std::range_error`:



## Public Member Functions

- **range\_error** (const [string](#) &\_\_arg)
- virtual const char \* **what** () const noexcept

## 4.939.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 130 of file `stdexcept`.

## 4.939.2 Member Function Documentation

4.939.2.1 virtual const char\* `std::runtime_error::what ( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

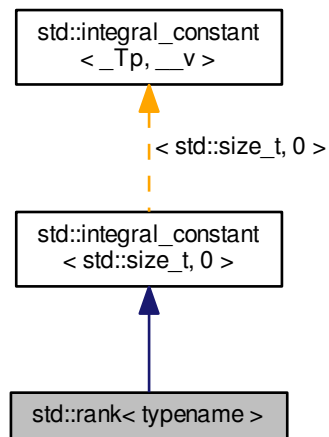
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

4.940 `std::rank< typename >` Struct Template Reference

Inheritance diagram for `std::rank< typename >`:



## Public Types

- typedef [integral\\_constant](#)  
`< std::size_t, __v >` **type**

- typedef `std::size_t` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** ()

#### Static Public Attributes

- static constexpr `std::size_t` **value**

##### 4.940.1 Detailed Description

`template<typename> struct std::rank< typename >`

rank

Definition at line 1243 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.941 `std::ratio< _Num, _Den >` Struct Template Reference

#### Public Types

- typedef [ratio](#)< num, den > **type**

#### Static Public Attributes

- static constexpr `intmax_t` **den**
- static constexpr `intmax_t` **num**

##### 4.941.1 Detailed Description

`template<intmax_t _Num, intmax_t _Den = 1> struct std::ratio< _Num, _Den >`

Provides compile-time rational arithmetic.

This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type `intmax_t`. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

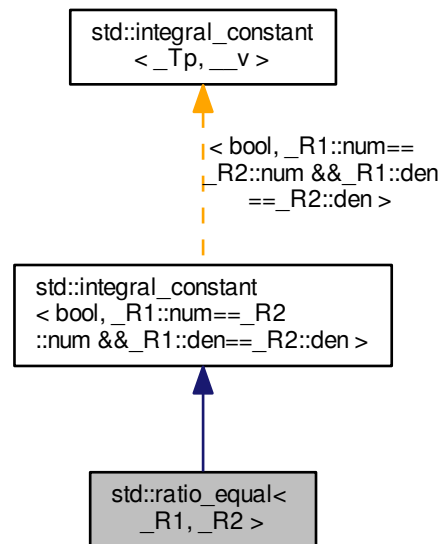
Definition at line 263 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 4.942 std::ratio\_equal&lt; \_R1, \_R2 &gt; Struct Template Reference

Inheritance diagram for std::ratio\_equal< \_R1, \_R2 >:



## Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- `constexpr operator value_type ()`

## Static Public Attributes

- `static constexpr bool` **value**

## 4.942.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_equal< _R1, _R2 >
```

ratio\_equal

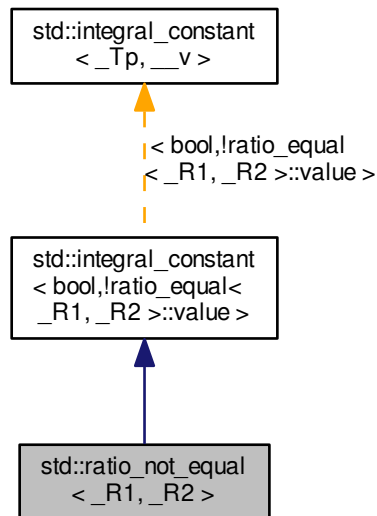
Definition at line 340 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 4.943 std::ratio\_not\_equal< \_R1, \_R2 > Struct Template Reference

Inheritance diagram for std::ratio\_not\_equal< \_R1, \_R2 >:



##### Public Types

- typedef [integral\\_constant](#) `< bool, __v > type`
- typedef bool **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** ()

##### Static Public Attributes

- static constexpr bool **value**

##### 4.943.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_not_equal< _R1, _R2 >
```

ratio\_not\_equal

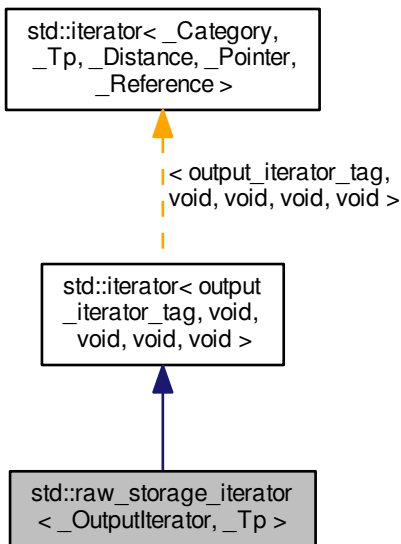
Definition at line 346 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 4.944 std::raw\_storage\_iterator< \_OutputIterator, \_Tp > Class Template Reference

Inheritance diagram for std::raw\_storage\_iterator< \_OutputIterator, \_Tp >:



#### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) `iterator_category`
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

#### Public Member Functions

- **raw\_storage\_iterator** (`_OutputIterator __x`)
- [raw\\_storage\\_iterator](#) & **operator\*** ()
- [raw\\_storage\\_iterator](#) `< _OutputIterator, _Tp >` & **operator++** ()
- [raw\\_storage\\_iterator](#) `< _OutputIterator, _Tp >` **operator++** (int)
- [raw\\_storage\\_iterator](#) & **operator=** (const `_Tp` & `__element`)

## Protected Attributes

- `_OutputIterator _M_iter`

## 4.944.1 Detailed Description

```
template<class _OutputIterator, class _Tp>class std::raw_storage_iterator< _OutputIterator, _Tp >
```

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 68 of file `stl_raw_storage_iter.h`.

## 4.944.2 Member Typedef Documentation

4.944.2.1 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type` [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.944.2.2 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.944.2.3 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` [inherited]

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.944.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference` [inherited]

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.944.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

4.945 `std::recursive_mutex` Class Reference

Inherits `std::__recursive_mutex_base`.

## Public Types

- `typedef __native_type * native_handle_type`

- **recursive\_mutex** (const **recursive\_mutex** &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** ()
- **recursive\_mutex** & **operator=** (const **recursive\_mutex** &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

- `typedef __gthread_recursive_mutex_t __native_type`

- `__native_type _M_mutex`

- `template<typename... _Args>  
result_of< _Tp &(_Args &&...)>  
::type operator() (_Args &&... __args) const`
- `reference_wrapper & operator= (const reference_wrapper< _Tp > &__inref) noexcept`

#### 4.946.1 Detailed Description

`template<typename _Tp> class std::reference_wrapper< _Tp >`

Primary class template for `reference_wrapper`.

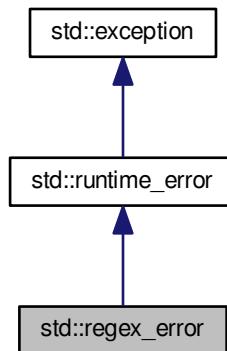
Definition at line 431 of file `functional`.

The documentation for this class was generated from the following file:

- `functional`

## 4.947 std::regex\_error Class Reference

Inheritance diagram for `std::regex_error`:



#### Public Member Functions

- `regex_error (regex_constants::error_type __ecode)`
- `regex_constants::error_type code () const`
- `virtual const char * what () const noexcept`

#### 4.947.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 136 of file `regex_error.h`.

## 4.947.2 Constructor &amp; Destructor Documentation

4.947.2.1 `std::regex_error::regex_error( regex_constants::error_type __ecode )` `[explicit]`

Constructs a `regex_error` object.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__ecode</code> | the regex error code. |
|----------------------|-----------------------|

## 4.947.3 Member Function Documentation

4.947.3.1 `regex_constants::error_type std::regex_error::code( ) const` `[inline]`

Gets the regex error code.

## Returns

the regex error code.

Definition at line 157 of file `regex_error.h`.

4.947.3.2 `virtual const char* std::runtime_error::what( ) const` `[virtual],[noexcept],[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

4.948 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

## Public Types

- `typedef std::ptrdiff_t difference_type`
- `typedef std::forward_iterator_tag iterator_category`
- `typedef const value_type * pointer`
- `typedef const value_type & reference`
- `typedef basic_regex< _Ch_type, _Rx_traits > regex_type`
- `typedef match_results< _Bi_iter > value_type`

## Public Member Functions

- `regex_iterator( )`
- `regex_iterator( _Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default)`
- `regex_iterator( const regex_iterator &__rhs)`
- `bool operator!= (const regex_iterator &__rhs)`
- `const value_type & operator* ( )`
- `regex_iterator & operator++ ( )`

- [regex\\_iterator operator++](#) (int)
- const [value\\_type](#) \* [operator->](#) ()
- [regex\\_iterator](#) & [operator=](#) (const [regex\\_iterator](#) & \_\_rhs)
- bool [operator==](#) (const [regex\\_iterator](#) & \_\_rhs)

#### 4.948.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2204 of file `regex.h`.

#### 4.948.2 Constructor & Destructor Documentation

4.948.2.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( )`

Provides a singular iterator, useful for indicating one-past-the-end of a range.

**Todo** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

4.948.2.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, regex_constants::match_flag_type __m = regex_constants::match_default )`

Constructs a `regex_iterator`...

##### Parameters

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <code>__a</code>  | [IN] The start of a text range to search.          |
| <code>__b</code>  | [IN] One-past-the-end of the text range to search. |
| <code>__re</code> | [IN] The regular expression to match.              |
| <code>__m</code>  | [IN] Policy flags for match rules.                 |

**Todo** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

4.948.2.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

Copy constructs a `regex_iterator`.

**Todo** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

## 4.948.3 Member Function Documentation

4.948.3.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!= ( const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

4.948.3.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ( )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

4.948.3.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ( )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

4.948.3.4 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ( int )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

4.948.3.5 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type* std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ( )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

4.948.3.6 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= ( const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

4.948.3.7 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator==( const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

**Todo** Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.949 `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef const `value_type` \* **pointer**
- typedef const `value_type` & **reference**
- typedef `basic_regex< _Ch_type, _Rx_traits >` **regex\_type**
- typedef `sub_match< _Bi_iter >` **value\_type**

### Public Member Functions

- `regex_token_iterator` ()
- `regex_token_iterator` (`_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, int \_\_submatch=0, `regex_constants::match_flag_type` \_\_m=`regex_constants::match_default`)
- `regex_token_iterator` (`_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, const `std::vector< int >` &\_\_submatches, `regex_constants::match_flag_type` \_\_m=`regex_constants::match_default`)
- template<`std::size_t` \_Nm>  
`regex_token_iterator` (`_Bi_iter` \_\_a, `_Bi_iter` \_\_b, const `regex_type` &\_\_re, const int(&\_\_submatches)[\_Nm], `regex_constants::match_flag_type` \_\_m=`regex_constants::match_default`)
- `regex_token_iterator` (const `regex_token_iterator` &\_\_rhs)
- bool `operator!=` (const `regex_token_iterator` &\_\_rhs)
- const `value_type` & `operator*` ()
- `regex_token_iterator` & `operator++` ()
- `regex_token_iterator` `operator++` (int)
- const `value_type` \* `operator->` ()
- `regex_token_iterator` & `operator=` (const `regex_token_iterator` &\_\_rhs)
- bool `operator==` (const `regex_token_iterator` &\_\_rhs)

### 4.949.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a `std::sub_match` object.

Definition at line 2318 of file `regex.h`.

## 4.949.2 Constructor &amp; Destructor Documentation

4.949.2.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( )`

Default constructs a regex\_token\_iterator.

**Todo** Implement this function.

A default-constructed regex\_token\_iterator is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

4.949.2.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, int __submatch = 0, regex_constants::match_flag_type __m = regex_constants::match_default )`

Constructs a regex\_token\_iterator...

## Parameters

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__a</code>        | [IN] The start of the text to search.                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>__b</code>        | [IN] One-past-the-end of the text to search.                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>__re</code>       | [IN] The regular expression to search for.                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>__submatch</code> | [IN] Which submatch to return. There are some special values for this parameter: <ul style="list-style-type: none"> <li>• -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)</li> <li>• 0 the entire string matching the subexpression is returned for each match within the text.</li> <li>• &gt;0 enumerates only the indicated subexpression from a match within the text.</li> </ul> |
| <code>__m</code>        | [IN] Policy flags for match rules.                                                                                                                                                                                                                                                                                                                                                                                                 |

**Todo** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

4.949.2.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, const std::vector< int > & __submatches, regex_constants::match_flag_type __m = regex_constants::match_default )`

Constructs a regex\_token\_iterator...

## Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

**Todo** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

4.949.2.4 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> template<std::size_t _Nm> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, const int(&) __submatches[_Nm], regex_constants::match_flag_type __m = regex_constants::match_default )`

Constructs a regex\_token\_iterator...

#### Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

**Todo** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html)

4.949.2.5 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

Copy constructs a regex\_token\_iterator.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__rhs</code> | [IN] A regex_token_iterator to copy. |
|--------------------|--------------------------------------|

**Todo** Implement this function.

### 4.949.3 Member Function Documentation

4.949.3.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=( const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

Compares a regex\_token\_iterator to another for inequality.

**Todo** Implement this function.

4.949.3.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ( )`

Dereferences a regex\_token\_iterator.

**Todo** Implement this function.

```
4.949.3.3 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
 regex_traits<_Ch_type>> regex_token_iterator& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
 >::operator++ ()
```

Increments a regex\_token\_iterator.

**Todo** Implement this function.

```
4.949.3.4 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
 = regex_traits<_Ch_type>> regex_token_iterator std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
 >::operator++ (int)
```

Postincrements a regex\_token\_iterator.

**Todo** Implement this function.

```
4.949.3.5 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
 = regex_traits<_Ch_type>> const value_type* std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
 >::operator-> ()
```

Selects a regex\_token\_iterator member.

**Todo** Implement this function.

```
4.949.3.6 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
 regex_traits<_Ch_type>> regex_token_iterator& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
 >::operator= (const regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs)
```

Assigns a regex\_token\_iterator to another.

Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__rhs</code> | [IN] A regex_token_iterator to copy. |
|--------------------|--------------------------------------|

**Todo** Implement this function.

```
4.949.3.7 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
 regex_traits<_Ch_type>> bool std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator== (const
 regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs)
```

Compares a regex\_token\_iterator to another for equality.

**Todo** Implement this function.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.950 std::regex\_traits<\_Ch\_type> Struct Template Reference

## Public Types

- typedef std::ctype\_base::mask **char\_class\_type**
- typedef \_Ch\_type **char\_type**
- typedef [std::locale](#) **locale\_type**
- typedef [std::basic\\_string](#)  
< char\_type > **string\_type**

## Public Member Functions

- [regex\\_traits](#) ()
- [locale\\_type](#) getloc () const
- [locale\\_type](#) imbue ([locale\\_type](#) \_\_loc)
- bool [isctype](#) (\_Ch\_type \_\_c, char\_class\_type \_\_f) const
- template<typename \_Fwd\_iter >  
char\_class\_type [lookup\\_classname](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last, bool \_\_icase=false) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [lookup\\_collatename](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [transform](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [transform\\_primary](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- char\_type [translate](#) (char\_type \_\_c) const
- char\_type [translate\\_nocase](#) (char\_type \_\_c) const
- int [value](#) (\_Ch\_type \_\_ch, int \_\_radix) const

## Static Public Member Functions

- static std::size\_t [length](#) (const char\_type \*\_\_p)

## Protected Attributes

- [locale\\_type](#) **\_M\_locale**

## 4.950.1 Detailed Description

template<typename \_Ch\_type>struct std::regex\_traits<\_Ch\_type>

Class regex\_traits. Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 51 of file regex.h.

## 4.950.2 Constructor &amp; Destructor Documentation

4.950.2.1 `template<typename _Ch_type > std::regex_traits< _Ch_type >::regex_traits ( ) [inline]`

Constructs a default traits object.

Definition at line 63 of file regex.h.

## 4.950.3 Member Function Documentation

4.950.3.1 `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::getloc ( ) const [inline]`

Gets a copy of the current locale in use by the regex\_traits object.

Definition at line 270 of file regex.h.

4.950.3.2 `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::imbue ( locale_type __loc ) [inline]`

Imbues the regex\_traits object with a copy of a new locale.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|

## Returns

a copy of the previous locale in use by the regex\_traits object.

## Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by \*this.

Definition at line 259 of file regex.h.

4.950.3.3 `template<typename _Ch_type > static std::size_t std::regex_traits< _Ch_type >::length ( const char_type * __p ) [inline], [static]`

Gives the length of a C-style string starting at \_\_p.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__p</code> | a pointer to the start of a character sequence. |
|------------------|-------------------------------------------------|

## Returns

the number of characters between \*\_\_p and the first default-initialized value of type char\_type. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 76 of file regex.h.

4.950.3.4 `template<typename _Ch_type > template<typename _Fwd_iter > char_class_type std::regex_traits< _Ch_type >::lookup_classname ( _Fwd_iter __first, _Fwd_iter __last, bool __icase = false ) const [inline]`

Maps one or more characters to a named character classification.

## Parameters

|                      |                                              |
|----------------------|----------------------------------------------|
| <code>__first</code> | beginning of the character sequence.         |
| <code>__last</code>  | one-past-the-end of the character sequence.  |
| <code>__icase</code> | ignores the case of the classification name. |

## Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- d
- w
- s
- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower
- print
- punct
- space
- upper
- xdigit

**Todo** Implement this function.

Definition at line 215 of file regex.h.

```
4.950.3.5 template<typename _Ch_type> template<typename _Fwd_iter> string_type std::regex_traits<_Ch_type>
 >::lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const [inline]
```

Gets a collation element by name.

## Parameters

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>__first</code> | beginning of the collation element name.        |
| <code>__last</code>  | one-past-the-end of the collation element name. |

**Returns**

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [`__first`, `__last`). Returns an empty string if the character sequence is not a valid collating element.

**Todo** Implement this function.

Definition at line 171 of file regex.h.

4.950.3.6 `template<typename _Ch_type> template<typename _Fwd_iter> string_type std::regex_traits<_Ch_type>::transform ( _Fwd_iter __first, _Fwd_iter __last ) const [inline]`

Gets a sort key for a character sequence.

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

Returns a sort key for the character sequence designated by the iterator range [`F1`, `F2`) such that if the character sequence [`G1`, `G2`) sorts before the character sequence [`H1`, `H2`) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

**Returns**

a locale-specific sort key equivalent to the input range.

**Exceptions**

|                            |                                                      |
|----------------------------|------------------------------------------------------|
| <code>std::bad_cast</code> | if the current locale does not have a collate facet. |
|----------------------------|------------------------------------------------------|

Definition at line 129 of file regex.h.

References `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

4.950.3.7 `template<typename _Ch_type> template<typename _Fwd_iter> string_type std::regex_traits<_Ch_type>::transform_primary ( _Fwd_iter __first, _Fwd_iter __last ) const [inline]`

Gets a sort key for a character sequence, independent of case.

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

Effects: if `typeid(use_facet<collate<_Ch_type>>) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

**Todo** Implement this function.

Definition at line 153 of file regex.h.

4.950.3.8 `template<typename _Ch_type> char_type std::regex_traits<_Ch_type>::translate ( char_type __c ) const`  
`[inline]`

Performs the identity translation.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

#### Returns

`__c`.

Definition at line 87 of file `regex.h`.

4.950.3.9 `template<typename _Ch_type> char_type std::regex_traits<_Ch_type>::translate_nocase ( char_type __c ) const`  
`[inline]`

Translates a character into a case-insensitive equivalent.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

#### Returns

the locale-specific lower-case equivalent of `__c`.

#### Exceptions

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <code>std::bad_cast</code> | if the imbued locale does not support the ctype facet. |
|----------------------------|--------------------------------------------------------|

Definition at line 100 of file `regex.h`.

The documentation for this struct was generated from the following file:

- [regex.h](#)

## 4.951 `std::remove_all_extents<_Tp>` Struct Template Reference

### Public Types

- `typedef _Tp type`

#### 4.951.1 Detailed Description

`template<typename _Tp> struct std::remove_all_extents<_Tp>`

`remove_all_extents`

Definition at line 1637 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.952 `std::remove_const< _Tp >` Struct Template Reference

### Public Types

- `typedef _Tp type`

#### 4.952.1 Detailed Description

```
template<typename _Tp>struct std::remove_const< _Tp >
```

`remove_const`

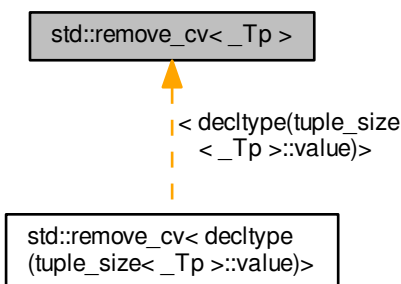
Definition at line 1327 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.953 `std::remove_cv< _Tp >` Struct Template Reference

Inheritance diagram for `std::remove_cv< _Tp >`:



### Public Types

- `typedef remove\_const< typename remove\_volatile< _Tp >::type >::type type`

#### 4.953.1 Detailed Description

`template<typename _Tp>struct std::remove_cv< _Tp >`

`remove_cv`

Definition at line 1345 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.954 `std::remove_extent< _Tp >` Struct Template Reference

Public Types

- `typedef _Tp type`

### 4.954.1 Detailed Description

`template<typename _Tp>struct std::remove_extent< _Tp >`

`remove_extent`

Definition at line 1624 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.955 `std::remove_pointer< _Tp >` Struct Template Reference

Inherits `std::__remove_pointer_helper< _Tp, typename >`.

Public Types

- `typedef _Tp type`

### 4.955.1 Detailed Description

`template<typename _Tp>struct std::remove_pointer< _Tp >`

`remove_pointer`

Definition at line 1661 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.956 `std::remove_reference< _Tp >` Struct Template Reference

Public Types

- `typedef _Tp type`

## 4.956.1 Detailed Description

```
template<typename _Tp>struct std::remove_reference< _Tp >
```

remove\_reference

Definition at line 1374 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.957 std::remove\_volatile&lt; \_Tp &gt; Struct Template Reference

## Public Types

- typedef `_Tp` **type**

## 4.957.1 Detailed Description

```
template<typename _Tp>struct std::remove_volatile< _Tp >
```

remove\_volatile

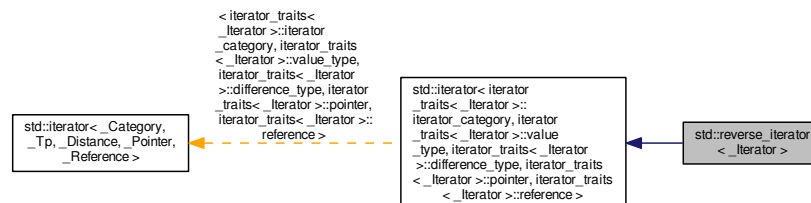
Definition at line 1336 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.958 std::reverse\_iterator&lt; \_Iterator &gt; Class Template Reference

Inheritance diagram for std::reverse\_iterator< \_Iterator >:



## Public Types

- typedef `__traits_type::difference_type` **difference\_type**
- typedef `iterator_traits< _Iterator >::iterator_category` **iterator\_category**

- typedef \_Iterator **iterator\_type**
- typedef \_\_traits\_type::pointer **pointer**
- typedef \_\_traits\_type::reference **reference**
- typedef iterator\_traits<\_Iterator>::value\_type **value\_type**

#### Public Member Functions

- [reverse\\_iterator](#) ()
- [reverse\\_iterator](#) (iterator\_type \_\_x)
- [reverse\\_iterator](#) (const [reverse\\_iterator](#) &\_\_x)
- template<typename \_Iter >  
[reverse\\_iterator](#) (const [reverse\\_iterator](#)<\_Iter> &\_\_x)
- iterator\_type [base](#) () const
- reference [operator\\*](#) () const
- [reverse\\_iterator](#) [operator+](#) (difference\_type \_\_n) const
- [reverse\\_iterator](#) & [operator++](#) ()
- [reverse\\_iterator](#) [operator++](#) (int)
- [reverse\\_iterator](#) & [operator+=](#) (difference\_type \_\_n)
- [reverse\\_iterator](#) [operator-](#) (difference\_type \_\_n) const
- [reverse\\_iterator](#) & [operator--](#) ()
- [reverse\\_iterator](#) [operator--](#) (int)
- [reverse\\_iterator](#) & [operator-=](#) (difference\_type \_\_n)
- pointer [operator->](#) () const
- reference [operator\[\]](#) (difference\_type \_\_n) const

#### Protected Types

- typedef iterator\_traits<\_Iterator> **\_\_traits\_type**

#### Protected Attributes

- \_Iterator **current**

#### 4.958.1 Detailed Description

```
template<typename _Iterator>class std::reverse_iterator<_Iterator>
```

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2*

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 96 of file `stl_iterator.h`.

## 4.958.2 Member Typedef Documentation

4.958.2.1 `typedef iterator_traits<_Iterator>::iterator_category std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference>::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.958.2.2 `typedef iterator_traits<_Iterator>::value_type std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference>::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

## 4.958.3 Constructor &amp; Destructor Documentation

4.958.3.1 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator( )` [inline]

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 120 of file `stl_iterator.h`.

Referenced by `std::reverse_iterator<_Iterator>::operator+()`, and `std::reverse_iterator<_Iterator>::operator-()`.

4.958.3.2 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator( iterator_type __x )` [inline], [explicit]

This iterator will move in the opposite direction that `x` does.

Definition at line 126 of file `stl_iterator.h`.

4.958.3.3 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator( const reverse_iterator<_Iterator> &__x )` [inline]

The copy constructor is normal.

Definition at line 131 of file `stl_iterator.h`.

4.958.3.4 `template<typename _Iterator> template<typename _Iter> std::reverse_iterator<_Iterator>::reverse_iterator( const reverse_iterator<_Iter> &__x )` [inline]

A `reverse_iterator` across other types can be copied if the underlying iterator can be converted to the type of `current`.

Definition at line 139 of file `stl_iterator.h`.

## 4.958.4 Member Function Documentation

4.958.4.1 `template<typename _Iterator> iterator_type std::reverse_iterator<_Iterator>::base( ) const` [inline]

Returns

`current`, the iterator used for underlying work.

Definition at line 146 of file `stl_iterator.h`.

Referenced by `std::operator==()`.

4.958.4.2 `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator*( ) const` `[inline]`

#### Returns

A reference to the value at `-current`

This requires that `-current` is dereferenceable.

#### Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

Definition at line 160 of file `stl_iterator.h`.

Referenced by `std::reverse_iterator<_Iterator>::operator->()`.

4.958.4.3 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator+ ( difference_type __n ) const` `[inline]`

#### Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 231 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::reverse_iterator()`.

4.958.4.4 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator++ ( )` `[inline]`

#### Returns

`*this`

Decrements the underlying iterator.

Definition at line 181 of file `stl_iterator.h`.

4.958.4.5 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator++ ( int )` `[inline]`

#### Returns

The original value of `*this`

Decrements the underlying iterator.

Definition at line 193 of file `stl_iterator.h`.

4.958.4.6 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator+= ( difference_type __n )` `[inline]`

#### Returns

`*this`

Moves the underlying iterator backwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 241 of file `stl_iterator.h`.

4.958.4.7 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator-( difference_type __n ) const [inline]`

#### Returns

A reverse\_iterator that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 253 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::reverse_iterator()`.

4.958.4.8 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator-- ( ) [inline]`

#### Returns

`*this`

Increments the underlying iterator.

Definition at line 206 of file `stl_iterator.h`.

4.958.4.9 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator-- ( int ) [inline]`

#### Returns

A reverse\_iterator with the previous value of `*this`

Increments the underlying iterator.

Definition at line 218 of file `stl_iterator.h`.

4.958.4.10 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator+=( difference_type __n ) [inline]`

#### Returns

`*this`

Moves the underlying iterator forwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 263 of file `stl_iterator.h`.

4.958.4.11 `template<typename _Iterator> pointer std::reverse_iterator<_Iterator>::operator-> ( ) const [inline]`

#### Returns

A pointer to the value at `-current`

This requires that `-current` is dereferenceable.

Definition at line 172 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::operator*()`.

4.958.4.12 `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator[] ( difference_type __n ) const [inline]`

## Returns

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

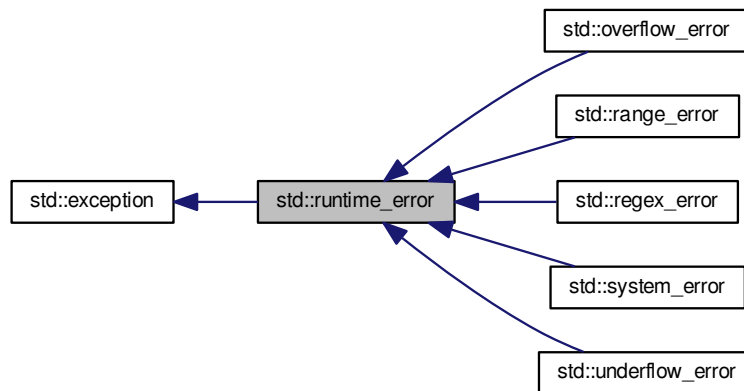
Definition at line 275 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 4.959 std::runtime\_error Class Reference

Inheritance diagram for `std::runtime_error`:



## Public Member Functions

- [runtime\\_error](#) (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

## 4.959.1 Detailed Description

One of two subclasses of exception.

Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 112 of file `stdexcept`.

## 4.959.2 Constructor &amp; Destructor Documentation

4.959.2.1 std::runtime\_error::runtime\_error ( const [string](#) &\_\_arg ) [explicit]

Takes a character string describing the error.

## 4.959.3 Member Function Documentation

4.959.3.1 `virtual const char* std::runtime_error::what ( ) const` `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

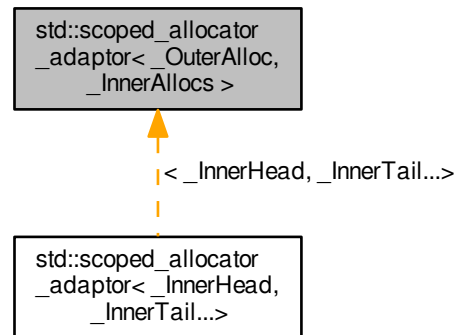
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

4.960 `std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >` Class Template Reference

Inheritance diagram for `std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >`:



## Public Types

- typedef `__traits::const_pointer` **const\_pointer**
- typedef `__traits::const_void_pointer` **const\_void\_pointer**
- typedef `__traits::difference_type` **difference\_type**
- typedef `__inner_type::__type` **inner\_allocator\_type**
- typedef `_OuterAlloc` **outer\_allocator\_type**
- typedef `__traits::pointer` **pointer**
- typedef `conditional< __any_of< __propagate_on_copy, _OuterAlloc, _InnerAllocs...>::value, true_type, false_type >`  
`::type` **propagate\_on\_container\_copy\_assignment**
- typedef `conditional< __any_of< __propagate_on_move, _OuterAlloc, _InnerAllocs...>::value, true_type, false_type >`  
`::type` **propagate\_on\_container\_move\_assignment**

- typedef [conditional](#)< \_\_any\_of  
< \_\_propagate\_on\_swap,  
\_OuterAlloc, \_InnerAllocs...>  
::value, [true\\_type](#), [false\\_type](#) >  
::type [propagate\\_on\\_container\\_swap](#)
- typedef [\\_\\_traits::size\\_type](#) [size\\_type](#)
- typedef [\\_\\_traits::value\\_type](#) [value\\_type](#)
- typedef [\\_\\_traits::void\\_pointer](#) [void\\_pointer](#)

## Public Member Functions

- template<typename \_Outer2 >  
**scoped\_allocator\_adaptor** (\_Outer2 && \_\_outer, const \_InnerAllocs &... \_\_inner)
- **scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#) &\_\_other)
- **scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#) &&\_\_other)
- template<typename \_Outer2 >  
**scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#)< \_Outer2, \_InnerAllocs...> &\_\_other)
- template<typename \_Outer2 >  
**scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#)< \_Outer2, \_InnerAllocs...> &&\_\_other)
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const\_void\_pointer \_\_hint)
- template<typename \_Tp, typename... \_Args>  
void **construct** (\_Tp \*\_\_p, \_Args &&... \_\_args)
- template<typename \_T1, typename \_T2, typename... \_Args1, typename... \_Args2>  
void **construct** ([pair](#)< \_T1, \_T2 > \*\_\_p, [piecewise\\_construct\\_t](#), [tuple](#)< \_Args1...> \_\_x, [tuple](#)< \_Args2...> \_\_y)
- template<typename \_T1, typename \_T2 >  
void **construct** ([pair](#)< \_T1, \_T2 > \*\_\_p)
- template<typename \_T1, typename \_T2, typename \_Up, typename \_Vp >  
void **construct** ([pair](#)< \_T1, \_T2 > \*\_\_p, \_Up &&\_\_u, \_Vp &&\_\_v)
- template<typename \_T1, typename \_T2, typename \_Up, typename \_Vp >  
void **construct** ([pair](#)< \_T1, \_T2 > \*\_\_p, const [pair](#)< \_Up, \_Vp > &\_\_x)
- template<typename \_T1, typename \_T2, typename \_Up, typename \_Vp >  
void **construct** ([pair](#)< \_T1, \_T2 > \*\_\_p, [pair](#)< \_Up, \_Vp > &&\_\_x)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Tp >  
void **destroy** (\_Tp \*\_\_p)
- inner\_allocator\_type & **inner\_allocator** () noexcept
- const inner\_allocator\_type & **inner\_allocator** () const noexcept
- size\_type **max\_size** () const
- outer\_allocator\_type & **outer\_allocator** () noexcept
- const outer\_allocator\_type & **outer\_allocator** () const noexcept
- [scoped\\_allocator\\_adaptor](#) **select\_on\_container\_copy\_construction** () const

## Friends

- template<typename \_Outer, typename... \_Inner>  
class **scoped\_allocator\_adaptor**
- template<typename... >  
class [\\_\\_inner\\_type\\_impl](#)
- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const [scoped\\_allocator\\_adaptor](#)< \_OutA1, \_InA...> &\_\_a, const [scoped\\_allocator\\_adaptor](#)< \_OutA2, \_InA...> &\_\_b) noexcept

## 4.960.1 Detailed Description

```
template<typename _OuterAlloc, typename... _InnerAllocs> class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
```

Primary class template.

Definition at line 176 of file `scoped_allocator`.

The documentation for this class was generated from the following file:

- [scoped\\_allocator](#)

## 4.961 std::seed\_seq Class Reference

## Public Types

- typedef uint\_least32\_t [result\\_type](#)

## Public Member Functions

- [seed\\_seq](#) ()
- template<typename \_IntType >  
  [seed\\_seq](#) (std::initializer\_list< \_IntType > il)
- template<typename \_InputIterator >  
  [seed\\_seq](#) (\_InputIterator \_\_begin, \_InputIterator \_\_end)
- template<typename \_RandomAccessIterator >  
  void **generate** (\_RandomAccessIterator \_\_begin, \_RandomAccessIterator \_\_end)
- template<typename OutputIterator >  
  void **param** (OutputIterator \_\_dest) const
- size\_t **size** () const

## 4.961.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 6024 of file `random.h`.

## 4.961.2 Member Typedef Documentation

## 4.961.2.1 typedef uint\_least32\_t std::seed\_seq::result\_type

The type of the seed vales.

Definition at line 6029 of file `random.h`.

## 4.961.3 Constructor &amp; Destructor Documentation

## 4.961.3.1 std::seed\_seq::seed\_seq( ) [inline]

Default constructor.

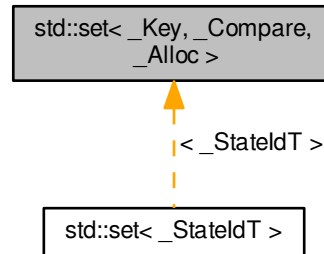
Definition at line 6032 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.962 std::set< \_Key, \_Compare, \_Alloc > Class Template Reference

Inheritance diagram for std::set< \_Key, \_Compare, \_Alloc >:



### Public Types

- typedef \_Key [key\\_type](#)
- typedef \_Key [value\\_type](#)
- typedef \_Compare [key\\_compare](#)
- typedef \_Compare [value\\_compare](#)
- typedef \_Alloc [allocator\\_type](#)
- typedef \_Key\_alloc\_type::pointer [pointer](#)
- typedef \_Key\_alloc\_type::const\_pointer [const\\_pointer](#)
- typedef \_Key\_alloc\_type::reference [reference](#)
- typedef \_Key\_alloc\_type::const\_reference [const\\_reference](#)
- typedef \_Rep\_type::const\_iterator [iterator](#)
- typedef \_Rep\_type::const\_iterator [const\\_iterator](#)
- typedef \_Rep\_type::const\_reverse\_iterator [reverse\\_iterator](#)
- typedef \_Rep\_type::const\_reverse\_iterator [const\\_reverse\\_iterator](#)
- typedef \_Rep\_type::size\_type [size\\_type](#)
- typedef \_Rep\_type::difference\_type [difference\\_type](#)

### Public Member Functions

- [set](#) ()
- [set](#) (const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())

- `template<typename _InputIterator >`  
`set` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `template<typename _InputIterator >`  
`set` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
- `set` (`const set &__x`)
- `set` (`set &&__x`) `noexcept(is_nothrow_copy_constructible<_Compare >`
- `set` (`initializer_list< value_type > __l`, `const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `iterator begin` () `const noexcept`
- `iterator cbegin` () `const noexcept`
- `iterator cend` () `const noexcept`
- `void clear` () `noexcept`
- `size_type count` (`const key_type &__x`) `const`
- `reverse_iterator crbegin` () `const noexcept`
- `reverse_iterator crend` () `const noexcept`
- `template<typename... _Args>`  
`std::pair< iterator, bool > emplace` (`_Args &&...__args`)
- `template<typename... _Args>`  
`iterator emplace_hint` (`const_iterator` \_\_pos, `_Args &&...__args`)
- `bool empty` () `const noexcept`
- `iterator end` () `const noexcept`
- `iterator erase` (`const_iterator` \_\_position)
- `size_type erase` (`const key_type &__x`)
- `iterator erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `allocator_type get_allocator` () `const noexcept`
- `std::pair< iterator, bool > insert` (`const value_type &__x`)
- `std::pair< iterator, bool > insert` (`value_type &&__x`)
- `iterator insert` (`const_iterator` \_\_position, `const value_type &__x`)
- `iterator insert` (`const_iterator` \_\_position, `value_type &&__x`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `void insert` (`initializer_list< value_type > __l`)
- `key_compare key_comp` () `const`
- `size_type max_size` () `const noexcept`
- `set & operator=` (`const set &__x`)
- `set & operator=` (`set &&__x`)
- `set & operator=` (`initializer_list< value_type > __l`)
- `reverse_iterator rbegin` () `const noexcept`
- `reverse_iterator rend` () `const noexcept`
- `size_type size` () `const noexcept`
- `void swap` (`set &__x`)
- `value_compare value_comp` () `const`
  
- `iterator find` (`const key_type &__x`)
- `const_iterator find` (`const key_type &__x`) `const`
  
- `iterator lower_bound` (`const key_type &__x`)
- `const_iterator lower_bound` (`const key_type &__x`) `const`
  
- `iterator upper_bound` (`const key_type &__x`)

- `const_iterator upper_bound` (const `key_type` &\_\_x) const
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const

#### Friends

- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`

#### 4.962.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set< _Key, _Compare, _Alloc >`

A standard container made up of unique keys, which can be retrieved in logarithmic time.

#### Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 90 of file `stl_set.h`.

#### 4.962.2 Member Typedef Documentation

4.962.2.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set< _Key, _Compare, _Alloc >::allocator_type`

Public typedefs.

Definition at line 107 of file `stl_set.h`.

4.962.2.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 128 of file `stl_set.h`.

4.962.2.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::const_pointer std::set< _Key, _Compare, _Alloc >::const_pointer`

Iterator-related typedefs.

Definition at line 121 of file stl\_set.h.

4.962.2.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::const_reference std::set< _Key, _Compare, _Alloc >::const_reference`

Iterator-related typedefs.

Definition at line 123 of file stl\_set.h.

4.962.2.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 130 of file stl\_set.h.

4.962.2.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type`

Iterator-related typedefs.

Definition at line 132 of file stl\_set.h.

4.962.2.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator`

Iterator-related typedefs.

Definition at line 127 of file stl\_set.h.

4.962.2.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Compare std::set< _Key, _Compare, _Alloc >::key_compare`

Public typedefs.

Definition at line 105 of file stl\_set.h.

4.962.2.9 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key std::set< _Key, _Compare, _Alloc >::key_type`

Public typedefs.

Definition at line 103 of file stl\_set.h.

4.962.2.10 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::pointer std::set< _Key, _Compare, _Alloc >::pointer`

Iterator-related typedefs.

Definition at line 120 of file stl\_set.h.

4.962.2.11 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::reference std::set< _Key, _Compare, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 122 of file stl\_set.h.

4.962.2.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc>::reverse_iterator`

Iterator-related typedefs.

Definition at line 129 of file stl\_set.h.

4.962.2.13 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::size_type std::set<_Key, _Compare, _Alloc>::size_type`

Iterator-related typedefs.

Definition at line 131 of file stl\_set.h.

4.962.2.14 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Compare std::set<_Key, _Compare, _Alloc>::value_compare`

Public typedefs.

Definition at line 106 of file stl\_set.h.

4.962.2.15 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef  
_Key std::set<_Key, _Compare, _Alloc>::value_type`

Public typedefs.

Definition at line 104 of file stl\_set.h.

#### 4.962.3 Constructor & Destructor Documentation

4.962.3.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::set<_Key, _Compare, _Alloc>::set ( ) [inline]`

Default constructor creates no elements.

Definition at line 139 of file stl\_set.h.

4.962.3.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::set<_Key, _Compare, _Alloc>::set ( const _Compare & __comp, const allocator_type & __a =  
allocator_type() ) [inline], [explicit]`

Creates a set with no elements.

##### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

Definition at line 148 of file stl\_set.h.

4.962.3.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename _InputIterator > std::set<_Key, _Compare, _Alloc>::set ( _InputIterator __first, _InputIterator  
__last ) [inline]`

Builds a set from a range.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a set consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 163 of file `stl_set.h`.

```
4.962.3.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 template<typename _InputIterator > std::set< _Key, _Compare, _Alloc >::set (_InputIterator __first, _InputIterator
 __last, const _Compare & __comp, const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from a range.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a set consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 180 of file `stl_set.h`.

```
4.962.3.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 std::set< _Key, _Compare, _Alloc >::set (const set< _Key, _Compare, _Alloc > & __x) [inline]
```

Set copy constructor.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A set of identical element and allocator types. |
|------------------|-------------------------------------------------|

The newly-created set uses a copy of the allocation object used by `__x`.

Definition at line 193 of file `stl_set.h`.

```
4.962.3.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 std::set< _Key, _Compare, _Alloc >::set (set< _Key, _Compare, _Alloc > && __x) [inline], [noexcept]
```

Set move constructor

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A set of identical element and allocator types. |
|------------------|-------------------------------------------------|

The newly-created set contains the exact contents of `x`. The contents of `x` are a valid, but unspecified set.

Definition at line 204 of file `stl_set.h`.

4.962.3.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::set<_Key, _Compare, _Alloc>::set ( initializer_list<value_type> & __l, const _Compare & __comp =  
_Compare(), const allocator_type & __a = allocator_type() ) [inline]`

Builds a set from an initializer\_list.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 218 of file `stl_set.h`.

#### 4.962.4 Member Function Documentation

4.962.4.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set<_Key, _Compare, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 298 of file `stl_set.h`.

4.962.4.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set<_Key, _Compare, _Alloc>::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 335 of file `stl_set.h`.

4.962.4.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set<_Key, _Compare, _Alloc>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 344 of file `stl_set.h`.

4.962.4.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::set<_Key, _Compare, _Alloc>::clear ( ) [inline], [noexcept]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 628 of file `stl_set.h`.

Referenced by `std::set<_StatIdT>::operator=()`.

4.962.4.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
size_type std::set<_Key, _Compare, _Alloc>::count ( const key_type & __x ) const [inline]`

Finds the number of elements.

## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

## Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 642 of file `stl_set.h`.

4.962.4.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::set<_Key, _Compare, _Alloc>::crbegin( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 353 of file `stl_set.h`.

4.962.4.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::set<_Key, _Compare, _Alloc>::crend( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 362 of file `stl_set.h`.

4.962.4.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename... _Args> std::pair<iterator, bool> std::set<_Key, _Compare, _Alloc>::emplace( _Args &&...  
_args ) [inline]`

Attempts to build and insert an element into the set.

## Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 413 of file `stl_set.h`.

4.962.4.9 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename... _Args> iterator std::set<_Key, _Compare, _Alloc>::emplace_hint( const_iterator __pos,  
_Args &&... _args ) [inline]`

Attempts to insert an element into the set.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 439 of file `stl_set.h`.

```
4.962.4.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool
 std::set<_Key, _Compare, _Alloc>::empty () const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 368 of file `stl_set.h`.

```
4.962.4.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
 std::set<_Key, _Compare, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 307 of file `stl_set.h`.

```
4.962.4.12 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 std::pair<iterator, iterator> std::set<_Key, _Compare, _Alloc>::equal_range (const key_type & __x)
 [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 722 of file `stl_set.h`.

```
4.962.4.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 std::pair<const_iterator, const_iterator> std::set<_Key, _Compare, _Alloc>::equal_range (const key_type
 & __x) const [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 726 of file stl\_set.h.

**4.962.4.14** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set< _Key, _Compare, _Alloc >::erase( const_iterator __position ) [inline]`

Erases an element from a set.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 550 of file stl\_set.h.

**4.962.4.15** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
size_type std::set< _Key, _Compare, _Alloc >::erase( const key_type & __x ) [inline]`

Erases elements according to the provided key.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

## Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 580 of file stl\_set.h.

4.962.4.16 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set<_Key, _Compare, _Alloc>::erase( const_iterator __first, const_iterator __last ) [inline]`

Erases a [`__first`,`__last`) range of elements from a set.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

#### Returns

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 601 of file `stl_set.h`.

4.962.4.17 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set<_Key, _Compare, _Alloc>::find( const key_type & __x ) [inline]`

Tries to locate an element in a set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 660 of file `stl_set.h`.

4.962.4.18 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::set<_Key, _Compare, _Alloc>::find( const key_type & __x ) const [inline]`

Tries to locate an element in a set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 664 of file `stl_set.h`.

4.962.4.19 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]`

Returns the allocator object with which the set was constructed.

Definition at line 289 of file `stl_set.h`.

4.962.4.20 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::insert ( const value_type & __x ) [inline]`

Attempts to insert an element into the set.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 460 of file `stl_set.h`.

Referenced by `std::set< _StateIDT >::operator=()`.

4.962.4.21 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set< _Key, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x ) [inline]`

Attempts to insert an element into the set.

#### Parameters

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 497 of file `stl_set.h`.

4.962.4.22 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
template<typename _InputIterator > void std::set< _Key, _Compare, _Alloc >::insert ( _InputIterator __first,  
_InputIterator __last ) [inline]`

A template function that attempts to insert a range of elements.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 517 of file `stl_set.h`.

**4.962.4.23** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::set< _Key, _Compare, _Alloc >::insert( initializer_list< value_type > __l ) [inline]`

Attempts to insert a list of elements into the set.

## Parameters

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
|------------------|------------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 529 of file `stl_set.h`.

Referenced by `std::set< _StatIdT >::insert()`.

**4.962.4.24** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
key_compare std::set< _Key, _Compare, _Alloc >::key_comp( ) const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 281 of file `stl_set.h`.

**4.962.4.25** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set< _Key, _Compare, _Alloc >::lower_bound( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 681 of file `stl_set.h`.

**4.962.4.26** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound( const key_type & __x ) const [inline]`

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 685 of file stl\_set.h.

```
4.962.4.27 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 size_type std::set< _Key, _Compare, _Alloc >::max_size() const [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 378 of file stl\_set.h.

```
4.962.4.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
 std::set< _Key, _Compare, _Alloc >::operator= (const set< _Key, _Compare, _Alloc > &__x) [inline]
```

Set assignment operator.

**Parameters**

|     |                                                 |
|-----|-------------------------------------------------|
| __x | A set of identical element and allocator types. |
|-----|-------------------------------------------------|

All the elements of \_\_x are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 233 of file stl\_set.h.

```
4.962.4.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
 std::set< _Key, _Compare, _Alloc >::operator= (set< _Key, _Compare, _Alloc > &&__x) [inline]
```

Set move assignment operator.

**Parameters**

|     |                                                 |
|-----|-------------------------------------------------|
| __x | A set of identical element and allocator types. |
|-----|-------------------------------------------------|

The contents of \_\_x are moved into this set (without copying). \_\_x is a valid, but unspecified set.

Definition at line 248 of file stl\_set.h.

```
4.962.4.30 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
 std::set< _Key, _Compare, _Alloc >::operator= (initializer_list< value_type > __l) [inline]
```

Set list assignment operator.

**Parameters**

|     |                      |
|-----|----------------------|
| __l | An initializer_list. |
|-----|----------------------|

This function fills a set with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 269 of file stl\_set.h.

4.962.4.31 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::set<_Key, _Compare, _Alloc>::rbegin( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 316 of file `stl_set.h`.

4.962.4.32 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
reverse_iterator std::set<_Key, _Compare, _Alloc>::rend( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 325 of file `stl_set.h`.

4.962.4.33 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
size_type std::set<_Key, _Compare, _Alloc>::size( ) const [inline], [noexcept]`

Returns the size of the set.

Definition at line 373 of file `stl_set.h`.

4.962.4.34 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void  
std::set<_Key, _Compare, _Alloc>::swap( set<_Key, _Compare, _Alloc> & __x ) [inline]`

Swaps data with another set.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A set of the same element and allocator types. |
|------------------|------------------------------------------------|

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 393 of file `stl_set.h`.

Referenced by `std::set<_StateldT>::operator=()`, and `std::swap()`.

4.962.4.35 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator  
std::set<_Key, _Compare, _Alloc>::upper_bound( const key_type & __x ) [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 697 of file `stl_set.h`.

4.962.4.36 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
const_iterator std::set<_Key, _Compare, _Alloc>::upper_bound( const key_type & __x ) const [inline]`

Finds the end of a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 701 of file `stl_set.h`.

4.962.4.37 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>`  
`value_compare std::set<_Key, _Compare, _Alloc>::value_comp ( ) const` `[inline]`

Returns the comparison object with which the set was constructed.

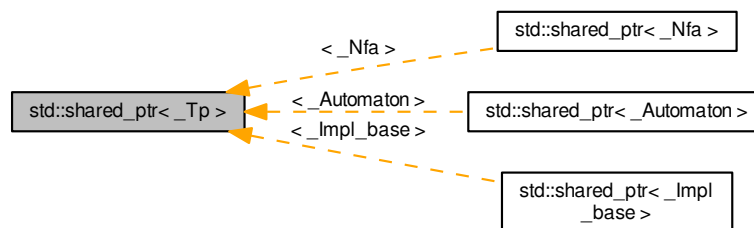
Definition at line 285 of file `stl_set.h`.

The documentation for this class was generated from the following file:

- [stl\\_set.h](#)

4.963 `std::shared_ptr<_Tp>` Class Template Reference

Inheritance diagram for `std::shared_ptr<_Tp>`:



## Public Types

- `typedef _Tp element_type`

## Public Member Functions

- `constexpr shared_ptr () noexcept`
- `shared_ptr (const shared_ptr &) noexcept=default`
- `template<typename _Tp1 >`  
`shared_ptr (_Tp1 *__p)`
- `template<typename _Tp1, typename _Deleter >`  
`shared_ptr (_Tp1 *__p, _Deleter __d)`
- `template<typename _Deleter >`  
`shared_ptr (nullptr_t __p, _Deleter __d)`

- `template<typename _Tp1, typename _Deleter, typename _Alloc >`  
`shared_ptr` (`_Tp1 *__p, _Deleter __d, _Alloc __a`)
- `template<typename _Deleter, typename _Alloc >`  
`shared_ptr` (`nullptr_t __p, _Deleter __d, _Alloc __a`)
- `template<typename _Tp1 >`  
`shared_ptr` (`const shared_ptr< _Tp1 > &__r, _Tp *__p`) `noexcept`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`  
`shared_ptr` (`const shared_ptr< _Tp1 > &__r`) `noexcept`
- `shared_ptr` (`shared_ptr &&__r`) `noexcept`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`  
`shared_ptr` (`shared_ptr< _Tp1 > &&__r`) `noexcept`
- `template<typename _Tp1 >`  
`shared_ptr` (`const weak_ptr< _Tp1 > &__r`)
- `template<typename _Tp1, typename _Del >`  
`shared_ptr` (`std::unique_ptr< _Tp1, _Del > &&__r`)
- `constexpr shared_ptr` (`nullptr_t __p`) `noexcept`
- `template<typename _Tp1 >`  
`shared_ptr` (`std::auto_ptr< _Tp1 > &&__r`)
- `_Tp * get ()` `const noexcept`
- `operator bool ()` `const`
- `std::add_lvalue_reference< _Tp >`  
`::type operator* ()` `const noexcept`
- `_Tp * operator-> ()` `const noexcept`
- `shared_ptr & operator=` (`const shared_ptr &`) `noexcept=default`
- `template<typename _Tp1 >`  
`shared_ptr & operator=` (`const shared_ptr< _Tp1 > &__r`) `noexcept`
- `shared_ptr & operator=` (`shared_ptr &&__r`) `noexcept`
- `template<class _Tp1 >`  
`shared_ptr & operator=` (`shared_ptr< _Tp1 > &&__r`) `noexcept`
- `template<typename _Tp1, typename _Del >`  
`shared_ptr & operator=` (`std::unique_ptr< _Tp1, _Del > &&__r`)
- `template<typename _Tp1 >`  
`bool owner_before` (`__shared_ptr< _Tp1, _Lp > const &__rhs`) `const`
- `template<typename _Tp1 >`  
`bool owner_before` (`__weak_ptr< _Tp1, _Lp > const &__rhs`) `const`
- `void reset ()` `noexcept`
- `template<typename _Tp1 >`  
`void reset` (`_Tp1 *__p`)
- `template<typename _Tp1, typename _Deleter >`  
`void reset` (`_Tp1 *__p, _Deleter __d`)
- `template<typename _Tp1, typename _Deleter, typename _Alloc >`  
`void reset` (`_Tp1 *__p, _Deleter __d, _Alloc __a`)
- `void swap` (`__shared_ptr< _Tp, _Lp > &__other`) `noexcept`
- `bool unique ()` `const noexcept`
- `long use_count ()` `const noexcept`

## Friends

- `template<typename _Tp1, typename _Alloc, typename... _Args>`  
`shared_ptr< _Tp1 > allocate_shared` (`const _Alloc &__a, _Args &&... __args`)

## 4.963.1 Detailed Description

```
template<typename _Tp>class std::shared_ptr< _Tp >
```

A smart pointer with reference-counted copy semantics.

The object pointed to is deleted when the last shared\_ptr pointing to it is destroyed or reset.

Definition at line 93 of file shared\_ptr.h.

## 4.963.2 Constructor &amp; Destructor Documentation

```
4.963.2.1 template<typename _Tp> constexpr std::shared_ptr< _Tp >::shared_ptr() [inline], [noexcept]
```

Construct an empty shared\_ptr.

## Postcondition

```
use_count()==0 && get()==0
```

Definition at line 100 of file shared\_ptr.h.

```
4.963.2.2 template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p)
[inline], [explicit]
```

Construct a shared\_ptr that owns the pointer \_\_p.

## Parameters

|            |                                                 |
|------------|-------------------------------------------------|
| <u>__p</u> | A pointer that is convertible to element_type*. |
|------------|-------------------------------------------------|

## Postcondition

```
use_count() == 1 && get() == __p
```

## Exceptions

|                            |                                  |
|----------------------------|----------------------------------|
| <i>std::bad_alloc</i> , in | which case delete __p is called. |
|----------------------------|----------------------------------|

Definition at line 112 of file shared\_ptr.h.

```
4.963.2.3 template<typename _Tp> template<typename _Tp1 , typename _Deleter > std::shared_ptr< _Tp >::shared_ptr (
_Tp1 * __p, _Deleter __d) [inline]
```

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.

## Parameters

|            |            |
|------------|------------|
| <u>__p</u> | A pointer. |
| <u>__d</u> | A deleter. |

## Postcondition

```
use_count() == 1 && get() == __p
```

## Exceptions

|                            |                                             |
|----------------------------|---------------------------------------------|
| <i>std::bad_alloc</i> , in | which case <code>__d(__p)</code> is called. |
|----------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 129 of file `shared_ptr.h`.

4.963.2.4 `template<typename _Tp> template<typename _Deleter > std::shared_ptr< _Tp >::shared_ptr( nullptr_t __p, _Deleter __d ) [inline]`

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

## Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__p</code> | A null pointer constant. |
| <code>__d</code> | A deleter.               |

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

|                            |                                             |
|----------------------------|---------------------------------------------|
| <i>std::bad_alloc</i> , in | which case <code>__d(__p)</code> is called. |
|----------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw

The last owner will call `__d(__p)`

Definition at line 146 of file `shared_ptr.h`.

4.963.2.5 `template<typename _Tp> template<typename _Tp1, typename _Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr( _Tp1 * __p, _Deleter __d, _Alloc __a ) [inline]`

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

## Parameters

|                  |               |
|------------------|---------------|
| <code>__p</code> | A pointer.    |
| <code>__d</code> | A deleter.    |
| <code>__a</code> | An allocator. |

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

|                            |                                             |
|----------------------------|---------------------------------------------|
| <i>std::bad_alloc</i> , in | which case <code>__d(__p)</code> is called. |
|----------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

\_\_shared\_ptr will release \_\_p by calling \_\_d(\_\_p)

Definition at line 165 of file shared\_ptr.h.

**4.963.2.6** `template<typename _Tp> template<typename _Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr ( nullptr_t __p, _Deleter __d, _Alloc __a ) [inline]`

Construct a shared\_ptr that owns a null pointer and the deleter \_\_d.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__p</code> | A null pointer constant. |
| <code>__d</code> | A deleter.               |
| <code>__a</code> | An allocator.            |

#### Postcondition

`use_count() == 1 && get() == __p`

#### Exceptions

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: \_Deleter's copy constructor and destructor must not throw \_Alloc's copy constructor and destructor must not throw.

The last owner will call `__d(__p)`

Definition at line 184 of file shared\_ptr.h.

**4.963.2.7** `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr ( const shared_ptr< _Tp1 > & __r, _Tp * __p ) [inline], [noexcept]`

Constructs a shared\_ptr instance that stores \_\_p and shares ownership with \_\_r.

#### Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__r</code> | A shared_ptr.                                         |
| <code>__p</code> | A pointer that will remain valid while *__r is valid. |

#### Postcondition

`get() == __p && use_count() == __r.use_count()`

This can be used to construct a shared\_ptr to a sub-object of an object managed by an existing shared\_ptr.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 206 of file shared\_ptr.h.

**4.963.2.8** `template<typename _Tp> template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> std::shared_ptr< _Tp >::shared_ptr ( const shared_ptr< _Tp1 > & __r ) [inline], [noexcept]`

If \_\_r is empty, constructs an empty shared\_ptr; otherwise construct a shared\_ptr that shares ownership with \_\_r.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__r</code> | A <code>shared_ptr</code> . |
|------------------|-----------------------------|

## Postcondition

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 218 of file `shared_ptr.h`.

**4.963.2.9** `template<typename _Tp> std::shared_ptr< _Tp >::shared_ptr ( shared_ptr< _Tp > && __r ) [inline], [noexcept]`

Move-constructs a `shared_ptr` instance from `__r`.

## Parameters

|                  |                                   |
|------------------|-----------------------------------|
| <code>__r</code> | A <code>shared_ptr</code> rvalue. |
|------------------|-----------------------------------|

## Postcondition

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 226 of file `shared_ptr.h`.

**4.963.2.10** `template<typename _Tp> template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> std::shared_ptr< _Tp >::shared_ptr ( shared_ptr< _Tp1 > && __r ) [inline], [noexcept]`

Move-constructs a `shared_ptr` instance from `__r`.

## Parameters

|                  |                                   |
|------------------|-----------------------------------|
| <code>__r</code> | A <code>shared_ptr</code> rvalue. |
|------------------|-----------------------------------|

## Postcondition

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 236 of file `shared_ptr.h`.

**4.963.2.11** `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr ( const weak_ptr< _Tp1 > & __r ) [inline], [explicit]`

Constructs a `shared_ptr` that shares ownership with `__r` and stores a copy of the pointer stored in `__r`.

## Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__r</code> | A <code>weak_ptr</code> . |
|------------------|---------------------------|

## Postcondition

`use_count() == __r.use_count()`

## Exceptions

|                     |                                                                                |
|---------------------|--------------------------------------------------------------------------------|
| <i>bad_weak_ptr</i> | when <code>__r.expired()</code> , in which case the constructor has no effect. |
|---------------------|--------------------------------------------------------------------------------|

Definition at line 248 of file `shared_ptr.h`.

4.963.2.12 `template<typename _Tp> constexpr std::shared_ptr< _Tp >::shared_ptr ( nullptr_t __p ) [inline],  
[noexcept]`

Construct an empty `shared_ptr`.

## Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__p</code> | A null pointer constant. |
|------------------|--------------------------|

## Postcondition

`use_count() == 0 && get() == nullptr`

Definition at line 265 of file `shared_ptr.h`.

## 4.963.3 Friends And Related Function Documentation

4.963.3.1 `template<typename _Tp> template<typename _Tp1, typename _Alloc, typename... _Args> shared_ptr<_Tp1>  
allocate_shared ( const _Alloc & __a, _Args &&... __args ) [friend]`

Create an object that is owned by a `shared_ptr`.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__a</code>    | An allocator.                                            |
| <code>__args</code> | Arguments for the <code>_Tp</code> object's constructor. |

## Returns

A `shared_ptr` that owns the newly created object.

## Exceptions

|           |                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------|
| <i>An</i> | exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> . |
|-----------|---------------------------------------------------------------------------------------------------|

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 595 of file `shared_ptr.h`.

The documentation for this class was generated from the following files:

- [shared\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 4.964 std::shuffle\_order\_engine&lt; \_RandomNumberEngine, \_\_k &gt; Class Template Reference

## Public Types

- typedef  
    `_RandomNumberEngine::result_type` [result\\_type](#)

## Public Member Functions

- [shuffle\\_order\\_engine](#) ()
- [shuffle\\_order\\_engine](#) (const `_RandomNumberEngine` &\_\_rng)
- [shuffle\\_order\\_engine](#) (`_RandomNumberEngine` &&\_\_rng)
- [shuffle\\_order\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`  
    [shuffle\\_order\\_engine](#) (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) operator() ()
- void [seed](#) ()
- void [seed](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` >  
    void [seed](#) (`_Sseq` &\_\_q)

## Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

## Static Public Attributes

- static constexpr size\_t [table\\_size](#)

## Friends

- template<typename `_RandomNumberEngine1` , size\_t \_\_k1, typename `_CharT` , typename `_Traits` >  
    [std::basic\\_ostream](#)< `_CharT`,  
    `_Traits` > & [operator<<](#) ([std::basic\\_ostream](#)< `_CharT`, `_Traits` > &\_\_os, const [std::shuffle\\_order\\_engine](#)< `_RandomNumberEngine1`, \_\_k1 > &\_\_x)
- bool [operator==](#) (const [shuffle\\_order\\_engine](#) &\_\_lhs, const [shuffle\\_order\\_engine](#) &\_\_rhs)
- template<typename `_RandomNumberEngine1` , size\_t \_\_k1, typename `_CharT` , typename `_Traits` >  
    [std::basic\\_istream](#)< `_CharT`,  
    `_Traits` > & [operator>>](#) ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &\_\_is, [std::shuffle\\_order\\_engine](#)< `_RandomNumberEngine1`, \_\_k1 > &\_\_x)

## 4.964.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __k>class std::shuffle_order_engine< _RandomNumberEngine, __k >`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.

Definition at line 1292 of file random.h.

## 4.964.2 Member Typedef Documentation

4.964.2.1 `template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type  
std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type`

The type of the generated random value.

Definition at line 1295 of file random.h.

## 4.964.3 Constructor &amp; Destructor Documentation

4.964.3.1 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( ) [inline]`

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

Definition at line 1308 of file random.h.

4.964.3.2 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( const _RandomNumberEngine & __rng ) [inline],[explicit]`

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

## Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1319 of file random.h.

4.964.3.3 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( _RandomNumberEngine && __rng ) [inline],[explicit]`

Move constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

## Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1330 of file random.h.

4.964.3.4 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( result_type __s ) [inline],[explicit]`

Seed constructs a shuffle\_order\_engine engine.

Constructs the underlying generator engine seeded with `__s`.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

Definition at line 1341 of file random.h.

```
4.964.3.5 template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq, typename =
typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq,
_RandomNumberEngine>::value> ::type> std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine(_Sseq & __q) [inline], [explicit]
```

Generator construct a shuffle\_order\_engine engine.

#### Parameters

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

Definition at line 1355 of file random.h.

#### 4.964.4 Member Function Documentation

```
4.964.4.1 template<typename _RandomNumberEngine, size_t __k> const _RandomNumberEngine&
std::shuffle_order_engine<_RandomNumberEngine, __k>::base() const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1398 of file random.h.

```
4.964.4.2 template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine<_RandomNumberEngine,
__k>::discard(unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 1419 of file random.h.

```
4.964.4.3 template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine<
_RandomNumberEngine, __k>::max() [inline], [static]
```

Gets the maximum value in the generated random number range.

Definition at line 1412 of file random.h.

References std::max().

```
4.964.4.4 template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine<
_RandomNumberEngine, __k>::min() [inline], [static]
```

Gets the minimum value in the generated random number range.

Definition at line 1405 of file random.h.

References std::min().

```
4.964.4.5 template<typename _RandomNumberEngine, size_t __k> shuffle_order_engine<_RandomNumberEngine, __k
>::result_type std::shuffle_order_engine<_RandomNumberEngine, __k>::operator()()
```

Gets the next value in the generated random number sequence.

Definition at line 818 of file bits/random.tcc.

```
4.964.4.6 template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine<_RandomNumberEngine,
__k>::seed() [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1364 of file random.h.

4.964.4.7 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine<_RandomNumberEngine, __k>::seed( result_type __s ) [inline]`

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1375 of file random.h.

4.964.4.8 `template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq > void std::shuffle_order_engine<_RandomNumberEngine, __k>::seed( _Sseq & __q ) [inline]`

Reseeds the shuffle\_order\_engine object with the given seed sequence.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__q</code> | A seed generator function. |
|------------------|----------------------------|

Definition at line 1388 of file random.h.

#### 4.964.5 Friends And Related Function Documentation

4.964.5.1 `template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const std::shuffle_order_engine<_RandomNumberEngine1, __k1> & __x ) [friend]`

Inserts the current state of a shuffle\_order\_engine random number generator engine `__x` into the output stream `__os`.

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.964.5.2 `template<typename _RandomNumberEngine, size_t __k> bool operator==( const shuffle_order_engine<_RandomNumberEngine, __k> & __lhs, const shuffle_order_engine<_RandomNumberEngine, __k> & __rhs ) [friend]`

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

#### Parameters

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__lhs</code> | A shuffle_order_engine random number generator object.       |
| <code>__rhs</code> | Another shuffle_order_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1443 of file random.h.

4.964.5.3 `template<typename _RandomNumberEngine, size_t _k> template<typename _RandomNumberEngine1, size_t _k1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & __is, std::shuffle_order_engine<_RandomNumberEngine1, _k1> & __x ) [friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__is</code> | An input stream.                                       |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

**Returns**

The input stream with the state of \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**4.965 std::slice Class Reference****Public Member Functions**

- [slice](#) ()
- [slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

**4.965.1 Detailed Description**

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 59 of file slice\_array.h.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 4.966 std::slice\_array&lt; \_Tp &gt; Class Template Reference

## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- `slice_array` (const `slice_array` &)
- void `operator%=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator%=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator&=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator&=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator*=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator*=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator+=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator+=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator-=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator-=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator/=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator/=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator<=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator<=>` (const \_Expr< \_Dom, \_Tp > &) const
- `slice_array` & `operator=` (const `slice_array` &)
- void `operator=` (const `valarray`< \_Tp > &) const
- void `operator=` (const \_Tp &) const
- template<class \_Dom >  
void `operator=` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator>=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator>=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator^=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator^=>` (const \_Expr< \_Dom, \_Tp > &) const
- void `operator|=>` (const `valarray`< \_Tp > &) const
- template<class \_Dom >  
void `operator|=>` (const \_Expr< \_Dom, \_Tp > &) const

## Friends

- class `valarray`< \_Tp >

## 4.966.1 Detailed Description

```
template<typename _Tp>class std::slice_array< _Tp >
```

Reference to one-dimensional subset of an array.

A `slice_array` is a reference to the actual elements of an array specified by a slice. The way to get a `slice_array` is to call `operator[](slice)` on a `valarray`. The returned `slice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `slice_array` refers to.

## Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

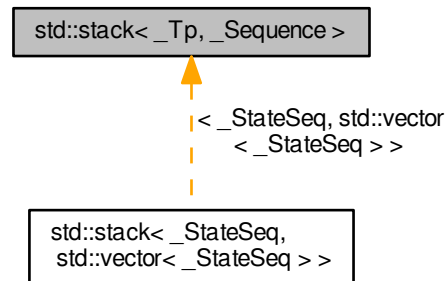
Definition at line 123 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

4.967 `std::stack<_Tp, _Sequence>` Class Template Reference

Inheritance diagram for `std::stack<_Tp, _Sequence>`:



## Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

## Public Member Functions

- [stack](#) (`const _Sequence &__c`)

- **stack** (\_Sequence &&\_\_c=\_Sequence())
- template<typename... \_Args>  
void **emplace** (\_Args &&...\_\_args)
- bool **empty** () const
- void **pop** ()
- void **push** (const value\_type &\_\_x)
- void **push** (value\_type &&\_\_x)
- size\_type **size** () const
- void **swap** (stack &\_\_s) noexcept(noexcept(swap(c
- **swap** (c, \_\_s.c)
- reference **top** ()
- const\_reference **top** () const

#### Public Attributes

- void \_\_s c

#### Protected Attributes

- \_Sequence c

#### Friends

- template<typename \_Tp1 , typename \_Seq1 >  
bool **operator**< (const stack< \_Tp1, \_Seq1 > &, const stack< \_Tp1, \_Seq1 > &)
- template<typename \_Tp1 , typename \_Seq1 >  
bool **operator==** (const stack< \_Tp1, \_Seq1 > &, const stack< \_Tp1, \_Seq1 > &)

#### 4.967.1 Detailed Description

template<typename \_Tp, typename \_Sequence = deque<\_Tp>>class std::stack< \_Tp, \_Sequence >

A standard container giving FILO behavior.

#### Template Parameters

|                        |                                                      |
|------------------------|------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                     |
| <code>_Sequence</code> | Type of underlying sequence, defaults to deque<_Tp>. |

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `back`, `push_back`, and `pop_front`, such as std::list, std::vector, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 96 of file `stl_stack.h`.

## 4.967.2 Constructor &amp; Destructor Documentation

4.967.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> std::stack< _Tp, _Sequence >::stack ( const _Sequence & __c ) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 134 of file `stl_stack.h`.

## 4.967.3 Member Function Documentation

4.967.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::stack< _Tp, _Sequence >::empty ( ) const [inline]`

Returns true if the stack is empty.

Definition at line 146 of file `stl_stack.h`.

4.967.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack< _Tp, _Sequence >::pop ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 212 of file `stl_stack.h`.

4.967.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack< _Tp, _Sequence >::push ( const value_type & __x ) [inline]`

Add data to the top of the stack.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 186 of file `stl_stack.h`.

4.967.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::stack< _Tp, _Sequence >::size ( ) const [inline]`

Returns the number of elements in the stack.

Definition at line 151 of file `stl_stack.h`.

4.967.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::stack< _Tp, _Sequence >::top ( ) [inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 159 of file `stl_stack.h`.

4.967.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::stack<_Tp, _Sequence>::top ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 170 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

## 4.968 `std::student_t_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **`student_t_distribution`** (`_RealType __n=_RealType(1)`)
- **`student_t_distribution`** (`const` [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &\_\_p)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, `_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, `_UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &\_\_p)
- [result\\_type](#) `max` () `const`
- [result\\_type](#) `min` () `const`
- `_RealType` `n` () `const`
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) `operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) `operator()` (`_UniformRandomNumberGenerator &__urng, const` [param\\_type](#) &\_\_p)
- [param\\_type](#) `param` () `const`
- `void` [param](#) (`const` [param\\_type](#) &\_\_param)
- `void` [reset](#) ()

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits > & operator<<` (`std::basic_ostream<_CharT, _Traits > &__os, const` `std::student_t_distribution<_RealType1 > &__x`)

- bool `operator==` (const `student_t_distribution` &\_\_d1, const `student_t_distribution` &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
`std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream< _CharT, _Traits > &__is`, `std::student_t_distribution< _RealType1 > &__x`)

#### 4.968.1 Detailed Description

template<typename \_RealType = double>class std::student\_t\_distribution< \_RealType >

A student\_t\_distribution random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3354 of file random.h.

#### 4.968.2 Member Typedef Documentation

4.968.2.1 template<typename \_RealType = double> typedef \_RealType std::student\_t\_distribution< \_RealType >::result\_type

The type of the range of the distribution.

Definition at line 3357 of file random.h.

#### 4.968.3 Member Function Documentation

4.968.3.1 template<typename \_RealType = double> result\_type std::student\_t\_distribution< \_RealType >::max ( ) const  
[inline]

Returns the least upper bound value of the distribution.

Definition at line 3437 of file random.h.

References std::numeric\_limits< \_Tp >::max().

4.968.3.2 template<typename \_RealType = double> result\_type std::student\_t\_distribution< \_RealType >::min ( ) const  
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3430 of file random.h.

References std::numeric\_limits< \_Tp >::min().

4.968.3.3 template<typename \_RealType = double> template<typename UniformRandomNumberGenerator > result\_type  
std::student\_t\_distribution< \_RealType >::operator() ( \_UniformRandomNumberGenerator & \_\_urng )  
[inline]

Generating functions.

Definition at line 3445 of file random.h.

References std::sqrt().

4.968.3.4 `template<typename _RealType = double> param_type std::student_t_distribution< _RealType >::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 3415 of file random.h.

4.968.3.5 `template<typename _RealType = double> void std::student_t_distribution< _RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3423 of file random.h.

4.968.3.6 `template<typename _RealType = double> void std::student_t_distribution< _RealType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3398 of file random.h.

References `std::normal_distribution< _RealType >::reset()`, and `std::gamma_distribution< _RealType >::reset()`.

#### 4.968.4 Friends And Related Function Documentation

4.968.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const  
std::student_t_distribution< _RealType1 > & __x ) [friend]`

Inserts a `student_t_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                 |
| <code>__x</code>  | A <code>student_t_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

4.968.4.2 `template<typename _RealType = double> bool operator==( const student_t_distribution< _RealType > & __d1,  
const student_t_distribution< _RealType > & __d2 ) [friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3494 of file random.h.

4.968.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits  
> std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::student_t_distribution< _RealType1 > & __x ) [friend]`

Extracts a `student_t_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <code>__is</code> | An input stream.                                         |
| <code>__x</code>  | A student_t_distribution random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.969 std::student\_t\_distribution&lt; \_RealType &gt;::param\_type Struct Reference

## Public Types

- typedef [student\\_t\\_distribution](#)  
< \_RealType > **distribution\_type**

## Public Member Functions

- **param\_type** ( \_RealType \_\_n=\_RealType(1))
- \_RealType **n** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.969.1 Detailed Description

```
template<typename _RealType = double>struct std::student_t_distribution< _RealType >::param_type
```

Parameter type.

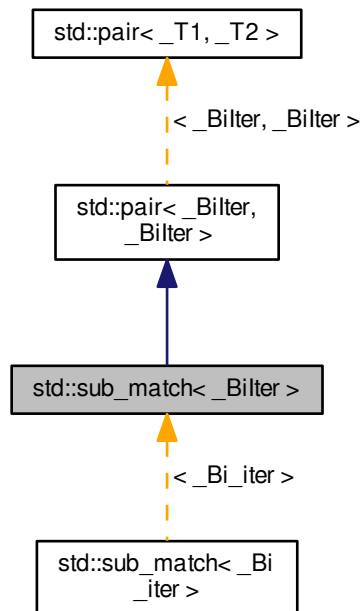
Definition at line 3363 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.970 std::sub\_match&lt;\_Bilter&gt; Class Template Reference

Inheritance diagram for std::sub\_match<\_Bilter>:



## Public Types

- typedef `__iter_traits::difference_type` **difference\_type**
- typedef `_Bilter` **first\_type**
- typedef `_Bilter` **iterator**
- typedef `_Bilter` **second\_type**
- typedef `std::basic_string`  
`< value_type >` **string\_type**
- typedef `__iter_traits::value_type` **value\_type**

## Public Member Functions

- int **compare** (const `sub_match` &\_\_s) const
- int **compare** (const `string_type` &\_\_s) const
- int **compare** (const `value_type` \*\_\_s) const
- `difference_type` **length** () const
- void `__p` **first** && **noexcept** (swap(`second`, \_\_p.second))
- `operator string_type` () const
- `string_type` **str** () const
- void **swap** (`pair` &\_\_p) noexcept(noexcept(swap(`first`

## Public Attributes

- [\\_Bilter first](#)
- [pair](#)  
[is\\_nothrow\\_move\\_assignable](#)  
[<\\_Bilter >::value first](#)
- bool **matched**
- [\\_Bilter second](#)
- **second**
- return \* **this**

## 4.970.1 Detailed Description

```
template<typename _Bilter> class std::sub_match< _Bilter >
```

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with std::basic\_string objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 741 of file regex.h.

## 4.970.2 Member Typedef Documentation

4.970.2.1 typedef [\\_Bilter](#) **std::pair<\_Bilter, \_Bilter>::second\_type** [inherited]

[first\\_type](#) is the first bound type

Definition at line 99 of file stl\_pair.h.

## 4.970.3 Member Function Documentation

4.970.3.1 `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const sub_match<_Bilter> & __s ) const`  
[inline]

Compares this and another matched sequence.

## Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <a href="#">__s</a> | Another matched sequence to compare to this one. |
|---------------------|--------------------------------------------------|

## Return values

|    |                                                                 |
|----|-----------------------------------------------------------------|
| <0 | this matched sequence will collate before <a href="#">__s</a> . |
| =0 | this matched sequence is equivalent to <a href="#">__s</a> .    |
| >0 | this matched sequence will collate after <a href="#">__s</a> .  |

Definition at line 802 of file regex.h.

Referenced by `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

4.970.3.2 `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const string_type & __s ) const`  
`[inline]`

Compares this sub\_match to a string.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A string to compare to this sub_match. |
|------------------|----------------------------------------|

#### Return values

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>&lt;0</code> | this matched sequence will collate before <code>__s</code> . |
| <code>=0</code>    | this matched sequence is equivalent to <code>__s</code> .    |
| <code>&gt;0</code> | this matched sequence will collate after <code>__s</code> .  |

Definition at line 815 of file regex.h.

4.970.3.3 `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const value_type * __s ) const` `[inline]`

Compares this sub\_match to a C-style string.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__s</code> | A C-style string to compare to this sub_match. |
|------------------|------------------------------------------------|

#### Return values

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>&lt;0</code> | this matched sequence will collate before <code>__s</code> . |
| <code>=0</code>    | this matched sequence is equivalent to <code>__s</code> .    |
| <code>&gt;0</code> | this matched sequence will collate after <code>__s</code> .  |

Definition at line 828 of file regex.h.

4.970.3.4 `template<typename _Bilter> difference_type std::sub_match<_Bilter>::length ( ) const` `[inline]`

Gets the length of the matching sequence.

Definition at line 759 of file regex.h.

4.970.3.5 `template<typename _Bilter> string_type std::sub_match<_Bilter>::operator string_type ( ) const` `[inline]`

Gets the matching sequence as a string.

#### Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the `str()` member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 772 of file regex.h.

4.970.3.6 `template<typename _Bilter> string_type std::sub_match<_Bilter>::str ( ) const` `[inline]`

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

Definition at line 785 of file regex.h.

Referenced by `std::sub_match<_Bi_iter>::compare()`, and `std::operator<<()`.

**4.970.4 Member Data Documentation****4.970.4.1 `_Bilter std::pair<_Bilter, _Bilter>::first` [inherited]**

`second_type` is the second bound type

Definition at line 101 of file stl\_pair.h.

Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator string_type()`, and `std::sub_match<_Bi_iter>::str()`.

**4.970.4.2 `_Bilter std::pair<_Bilter, _Bilter>::second` [inherited]**

`first` is a copy of the first object

Definition at line 102 of file stl\_pair.h.

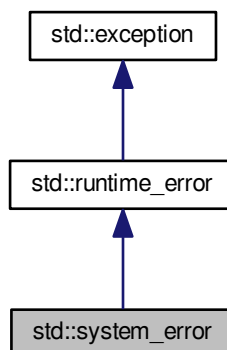
Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator string_type()`, and `std::sub_match<_Bi_iter>::str()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

**4.971 std::system\_error Class Reference**

Inheritance diagram for `std::system_error`:



## Public Member Functions

- **system\_error** ([error\\_code](#) \_\_ec=[error\\_code](#)())
- **system\_error** ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat, const [string](#) &\_\_what)
- const [error\\_code](#) & **code** () const noexcept
- virtual const char \* **what** () const noexcept

## 4.971.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 309 of file `system_error`.

## 4.971.2 Member Function Documentation

4.971.2.1 `virtual const char* std::runtime_error::what ( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

4.972 `std::thread` Class Reference

## Classes

- class [id](#)  
*thread::id*

## Public Types

- typedef [shared\\_ptr](#)< [\\_Impl\\_base](#) > **\_\_shared\_base\_type**
- typedef [\\_\\_gthread\\_t](#) **native\_handle\_type**

## Public Member Functions

- **thread** ([thread](#) &)=delete
- **thread** (const [thread](#) &)=delete
- **thread** ([thread](#) &&\_\_t) noexcept
- template<typename [\\_Callable](#) , typename... [\\_Args](#)>  
**thread** ([\\_Callable](#) &&\_\_f, [\\_Args](#) &&...\_\_args)
- void **detach** ()
- [thread::id](#) **get\_id** () const noexcept
- void **join** ()
- bool **joinable** () const noexcept
- native\_handle\_type [native\\_handle](#) ()

- `thread` & `operator=` (const `thread` &)=delete
- `thread` & `operator=` (`thread` &&\_\_t) noexcept
- void `swap` (`thread` &\_\_t) noexcept

#### Static Public Member Functions

- static unsigned int `hardware_concurrency` () noexcept

#### 4.972.1 Detailed Description

`thread`

Definition at line 60 of file `thread`.

#### 4.972.2 Member Function Documentation

##### 4.972.2.1 `native_handle_type` `std::thread::native_handle` ( ) [inline]

#### Precondition

`thread` is joinable

Definition at line 177 of file `thread`.

The documentation for this class was generated from the following file:

- `thread`

## 4.973 `std::thread::id` Class Reference

#### Public Member Functions

- `id` (native\_handle\_type \_\_id)

#### Friends

- class `hash`< `thread::id` >
- bool `operator`< (`thread::id` \_\_x, `thread::id` \_\_y) noexcept
- template<class `_CharT`, class `_Traits` >  
`basic_ostream`< `_CharT`, `_Traits` > & `operator`<< (`basic_ostream`< `_CharT`, `_Traits` > &\_\_out, `thread::id` \_\_id)
- bool `operator`== (`thread::id` \_\_x, `thread::id` \_\_y) noexcept
- class `thread`

#### 4.973.1 Detailed Description

`thread::id`

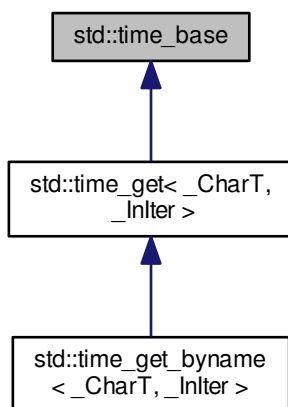
Definition at line 68 of file `thread`.

The documentation for this class was generated from the following file:

- `thread`

## 4.974 std::time\_base Class Reference

Inheritance diagram for std::time\_base:



#### Public Types

- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }

#### 4.974.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

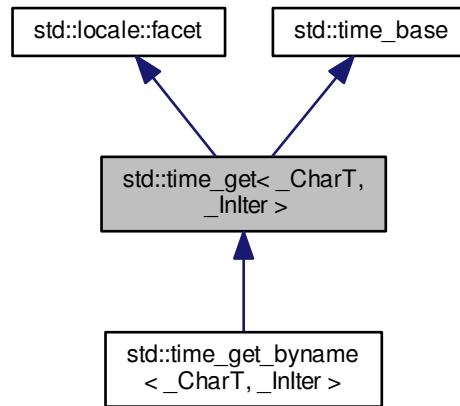
Definition at line 52 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.975 std::time\_get&lt; \_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for std::time\_get< \_CharT, \_InIter >:



## Public Types

- typedef `basic_string< _CharT >` `__string_type`
- enum `dateorder` {  
  `no_order`, `dmy`, `mdy`, `ymd`,  
  `ydm` }
- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

## Public Member Functions

- `time_get` (`size_t` \_\_refs=0)
- `dateorder` `date_order` () const
- `iter_type` `get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_monthname` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_time` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_weekday` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const
- `iter_type` `get_year` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \* \_\_tm) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~time_get()`
- `iter_type M_extract_name(iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type M_extract_num(iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type M_extract_via_format(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const`
- `iter_type M_extract_wday_or_month(iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- virtual `dateorder do_date_order() const`
- virtual `iter_type do_get_date(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_monthname(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_time(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_weekday(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_year(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale(_c_locale &__cloc) throw()`
- static void `_S_create_c_locale(_c_locale &__cloc, const char *__s, _c_locale __old=0)`
- static void `_S_destroy_c_locale(_c_locale &__cloc)`
- static `_c_locale _S_get_c_locale()`
- static const char \* `_S_get_c_name() throw()`
- static `_c_locale _S_lc_ctype_c_locale(_c_locale __cloc, const char *__s)`

## 4.975.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get<_CharT, _InIter>`

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The `time_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_get` facet.

Definition at line 368 of file `locale_facets_nonio.h`.

## 4.975.2 Member Typedef Documentation

4.975.2.1 `template<typename _CharT, typename _InIter> typedef _CharT std::time_get<_CharT, _InIter>::char_type`

Public typedefs.

Definition at line 374 of file `locale_facets_nonio.h`.

4.975.2.2 `template<typename _CharT, typename _Inlter > typedef _Inlter std::time_get< _CharT, _Inlter >::iter_type`

Public typedefs.

Definition at line 375 of file locale\_facets\_nonio.h.

#### 4.975.3 Constructor & Destructor Documentation

4.975.3.1 `template<typename _CharT, typename _Inlter > std::time_get< _CharT, _Inlter >::time_get ( size_t __refs = 0 )`  
`[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 390 of file locale\_facets\_nonio.h.

4.975.3.2 `template<typename _CharT, typename _Inlter > virtual std::time_get< _CharT, _Inlter >::~time_get ( )`  
`[inline], [protected], [virtual]`

Destructor.

Definition at line 546 of file locale\_facets\_nonio.h.

#### 4.975.4 Member Function Documentation

4.975.4.1 `template<typename _CharT, typename _Inlter > dateorder std::time_get< _CharT, _Inlter >::date_order ( ) const`  
`[inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

##### Returns

A member of `timebase::dateorder`.

Definition at line 407 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _Inlter >::do_date_order()`.

4.975.4.2 `template<typename _CharT, typename _Inlter > time_base::dateorder std::time_get< _CharT, _Inlter >::do_date_order ( ) const`  
`[protected], [virtual]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

**Returns**

A member of timebase::dateorder.

Definition at line 620 of file locale\_facets\_nonio.tcc.

Referenced by std::time\_get< \_CharT, \_InIter >::date\_order().

4.975.4.3 `template<typename _CharT, typename _InIter> _InIter std::time_get< _CharT, _InIter >::do_get_date ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected], [virtual]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_date() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond date string.

Definition at line 1047 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_date().

4.975.4.4 `template<typename _CharT, typename _InIter> _InIter std::time_get< _CharT, _InIter >::do_get_monthname ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected], [virtual]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_monthname() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

Definition at line 1092 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_monthname().

**4.975.4.5** `template<typename _CharT, typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_time ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected],  
[virtual]`

Parse input time string.

This function parses a time according to the format x and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_time() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

Definition at line 1030 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_time().

**4.975.4.6** `template<typename _CharT, typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected],  
[virtual]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_weekday() for details.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| <code>__beg</code> | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |

|                    |                                  |
|--------------------|----------------------------------|
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

Definition at line 1064 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_weekday().

```
4.975.4.7 template<typename _CharT, typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_year (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [protected],
 [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_year() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

Definition at line 1120 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_year().

```
4.975.4.8 template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_date (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_date().

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond date string.

Definition at line 456 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_date()`.

**4.975.4.9** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_monthname ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_monthname()`.

**4.975.4.10** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_time ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]`

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

Definition at line 431 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_time()`.

**4.975.4.11** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_weekday ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

Definition at line 484 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_weekday()`.

**4.975.4.12** `template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_year ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond year.

Definition at line 539 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_year()`.

## 4.975.5 Member Data Documentation

4.975.5.1 `template<typename _CharT, typename _InIter> locale::id std::time_get< _CharT, _InIter >::id` `[static]`

Numpunct facet id.

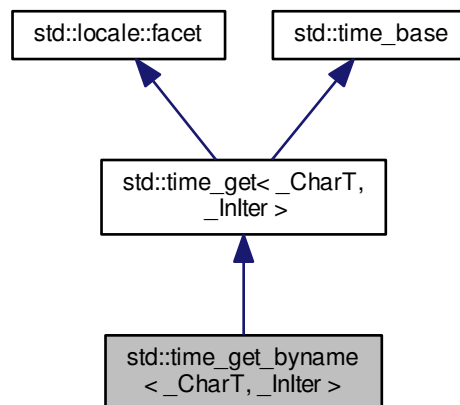
Definition at line 380 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 4.976 std::time\_get\_byname&lt; \_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for `std::time_get_byname< _CharT, _InIter >`:



## Public Types

- typedef [basic\\_string](#)<\_CharT> **\_\_string\_type**
- typedef \_CharT **char\_type**
- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }
- typedef \_InIter **iter\_type**

## Public Member Functions

- **time\_get\_byname** (const char \*, size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const
- [iter\\_type](#) **get\_date** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) **get\_monthname** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) **get\_time** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) **get\_weekday** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) **get\_year** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- [iter\\_type](#) **M\_extract\_name** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type](#) **M\_extract\_num** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type](#) **M\_extract\_via\_format** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format) const
- [iter\\_type](#) **M\_extract\_wday\_or\_month** ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- virtual dateorder [do\\_date\\_order](#) () const
- virtual [iter\\_type](#) [do\\_get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.976.1 Detailed Description

```
template<typename _CharT, typename _InIter>class std::time_get_byname< _CharT, _InIter >
```

class time\_get\_byname [22.2.5.2].

Definition at line 686 of file locale\_facets\_nonio.h.

## 4.976.2 Member Function Documentation

**4.976.2.1** template<typename \_CharT, typename \_InIter > dateorder std::time\_get< \_CharT, \_InIter >::date\_order ( ) const  
[inline], [inherited]

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format x given to time\_put::put() only uses month, day, and year. If the format x for the associated locale uses other fields, this function returns timebase::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

## Returns

A member of timebase::dateorder.

Definition at line 407 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_date\_order().

**4.976.2.2** template<typename \_CharT, typename \_InIter > time\_base::dateorder std::time\_get< \_CharT, \_InIter >::do\_date\_order ( ) const [protected], [virtual], [inherited]

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format x given to time\_put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

## Returns

A member of timebase::dateorder.

Definition at line 620 of file locale\_facets\_nonio.tcc.

Referenced by std::time\_get< \_CharT, \_InIter >::date\_order().

4.976.2.3 `template<typename _CharT, typename _InIter> _InIter std::time_get<_CharT, _InIter>::do_get_date ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const` [protected], [virtual], [inherited]

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See Also

`get_date()` for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond date string.

Definition at line 1047 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_date()`.

4.976.2.4 `template<typename _CharT, typename _InIter> _InIter std::time_get<_CharT, _InIter>::do_get_monthname ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const` [protected], [virtual], [inherited]

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See Also

`get_monthname()` for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

Definition at line 1092 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_monthname().

**4.976.2.5** `template<typename _CharT, typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_time ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected],  
[virtual], [inherited]`

Parse input time string.

This function parses a time according to the format x and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_time() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

Definition at line 1030 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_time().

**4.976.2.6** `template<typename _CharT, typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected],  
[virtual], [inherited]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_weekday() for details.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| <code>__beg</code> | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |

|                    |                                  |
|--------------------|----------------------------------|
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

Definition at line 1064 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_weekday().

**4.976.2.7** `template<typename _CharT, typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_year ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected],  
[virtual], [inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See Also**

get\_year() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

Definition at line 1120 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_year().

**4.976.2.8** `template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_date ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline],  
[inherited]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_date().

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, *err* |= ios\_base::failbit. If parsing reads all the characters, *err* |= ios\_base::eofbit.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond date string.

Definition at line 456 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_date()`.

```
4.976.2.9 template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_monthname (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline],
 [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_monthname()`.

```
4.976.2.10 template<typename _CharT, typename _InIter> iter_type std::time_get< _CharT, _InIter >::get_time (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline],
 [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

Definition at line 431 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_time()`.

```
4.976.2.11 template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_weekday (
 iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline],
 [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

Definition at line 484 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_weekday()`.

```
4.976.2.12 template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_year (iter_type
 __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline],
 [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond year.

Definition at line 539 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _Inlter >::do_get_year()`.

## 4.976.3 Member Data Documentation

4.976.3.1 `template<typename _CharT, typename _Inlter > locale::id std::time_get< _CharT, _Inlter >::id` `[static]`, `[inherited]`

Numpunct facet id.

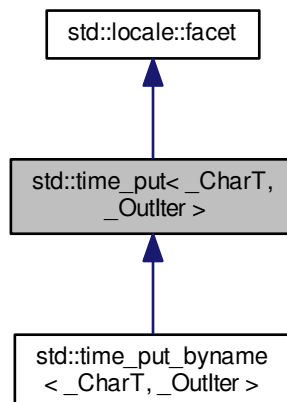
Definition at line 380 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.977 std::time\_put&lt; \_CharT, \_Outlter &gt; Class Template Reference

Inheritance diagram for `std::time_put< _CharT, _Outlter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_Outlter` `iter_type`

## Public Member Functions

- `time_put` (`size_t` \_\_refs=0)
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, const `tm` \*\_\_tm, const `_CharT` \*\_\_beg, const `_CharT` \*\_\_end) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, const `tm` \*\_\_tm, `char` \_\_format, `char` \_\_mod=0) const

## Static Public Attributes

- static `locale::id` id

## Protected Member Functions

- virtual `~time_put` ()
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, const `tm` \*\_\_tm, `char` \_\_format, `char` \_\_mod) const

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (`_c_locale` &\_\_cloc) throw ()
- static void `_S_create_c_locale` (`_c_locale` &\_\_cloc, const `char` \*\_\_s, `_c_locale` \_\_old=0)
- static void `_S_destroy_c_locale` (`_c_locale` &\_\_cloc)
- static `_c_locale _S_get_c_locale` ()
- static const `char` \* `_S_get_c_name` () throw ()
- static `_c_locale _S_lc_ctype_c_locale` (`_c_locale` \_\_cloc, const `char` \*\_\_s)

## 4.977.1 Detailed Description

template<typename `_CharT`, typename `_Outlter`>class std::time\_put< `_CharT`, `_Outlter` >

Primary class template `time_put`.

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

The `time_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_put` facet.

Definition at line 715 of file `locale_facets_nonio.h`.

## 4.977.2 Member Typedef Documentation

4.977.2.1 template<typename `_CharT`, typename `_Outlter` > typedef `_CharT` std::time\_put< `_CharT`, `_Outlter` >::char\_type

Public typedefs.

Definition at line 721 of file `locale_facets_nonio.h`.

4.977.2.2 `template<typename _CharT, typename _Outlter > typedef _Outlter std::time_put<_CharT, _Outlter >::iter_type`

Public typedefs.

Definition at line 722 of file locale\_facets\_nonio.h.

#### 4.977.3 Constructor & Destructor Documentation

4.977.3.1 `template<typename _CharT, typename _Outlter > std::time_put<_CharT, _Outlter >::time_put ( size_t __refs = 0 )  
[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 736 of file locale\_facets\_nonio.h.

4.977.3.2 `template<typename _CharT, typename _Outlter > virtual std::time_put<_CharT, _Outlter >::~time_put ( )  
[inline], [protected], [virtual]`

Destructor.

Definition at line 782 of file locale\_facets\_nonio.h.

#### 4.977.4 Member Function Documentation

4.977.4.1 `template<typename _CharT, typename _Outlter > _Outlter std::time_put<_CharT, _Outlter >::do_put ( iter_type  
__s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod ) const [protected],  
[virtual]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

##### See Also

`put()` for more details.

##### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | <code>char_type</code> to use for padding.   |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

Definition at line 1178 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

Referenced by std::time\_put< \_CharT, \_Outlter >::put().

**4.977.4.2** template<typename \_CharT, typename \_Outlter > \_Outlter std::time\_put< \_CharT, \_Outlter >::put ( iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const tm \* \_\_tm, const \_CharT \* \_\_beg, const \_CharT \* \_\_end ) const

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

**Parameters**

|        |                                              |
|--------|----------------------------------------------|
| __s    | The stream to write to.                      |
| __io   | Source of locale.                            |
| __fill | char_type to use for padding.                |
| __tm   | Struct tm with date and time info to format. |
| __beg  | Start of format string.                      |
| __end  | End of format string.                        |

**Returns**

Iterator after writing.

Definition at line 1143 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::narrow().

**4.977.4.3** template<typename \_CharT, typename \_Outlter > iter\_type std::time\_put< \_CharT, \_Outlter >::put ( iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const tm \* \_\_tm, char \_\_format, char \_\_mod=0 ) const [inline]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

**Parameters**

|          |                                              |
|----------|----------------------------------------------|
| __s      | The stream to write to.                      |
| __io     | Source of locale.                            |
| __fill   | char_type to use for padding.                |
| __tm     | Struct tm with date and time info to format. |
| __format | Format char.                                 |
| __mod    | Optional modifier char.                      |

**Returns**

Iterator after writing.

Definition at line 775 of file locale\_facets\_nonio.h.

References std::time\_put< \_CharT, \_Outlter >::do\_put().

## 4.977.5 Member Data Documentation

4.977.5.1 `template<typename _CharT, typename _Outlter > locale::id std::time_put< _CharT, _Outlter >::id` `[static]`

Numpunct facet id.

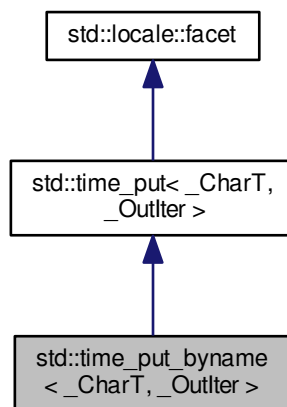
Definition at line 726 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 4.978 std::time\_put\_byname&lt; \_CharT, \_Outlter &gt; Class Template Reference

Inheritance diagram for `std::time_put_byname< _CharT, _Outlter >`:



## Public Types

- `typedef _CharT char_type`
- `typedef _Outlter iter_type`

## Public Member Functions

- `time_put_byname` (`const char *`, `size_t __refs=0`)
- `iter_type put` (`iter_type __s`, `ios_base & __io`, `char_type __fill`, `const tm * __tm`, `const _CharT * __beg`, `const _CharT * __end`) `const`
- `iter_type put` (`iter_type __s`, `ios_base & __io`, `char_type __fill`, `const tm * __tm`, `char __format`, `char __mod=0`) `const`

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [iter\\_type](#) [do\\_put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.978.1 Detailed Description

```
template<typename _CharT, typename _Outiter>class std::time_put_byname< _CharT, _Outiter >
```

class time\_put\_byname [22.2.5.4].

Definition at line 811 of file locale\_facets\_nonio.h.

## 4.978.2 Member Function Documentation

4.978.2.1 `template<typename _CharT, typename _Outiter> _Outiter std::time_put< _CharT, _Outiter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod ) const` [protected], [virtual], [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

## See Also

[put\(\)](#) for more details.

## Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | <code>char_type</code> to use for padding.   |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

Definition at line 1178 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

Referenced by std::time\_put< \_CharT, \_Outlter >::put().

**4.978.2.2** template<typename \_CharT, typename \_Outlter > \_Outlter std::time\_put< \_CharT, \_Outlter >::put ( iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const tm \* \_\_tm, const \_CharT \* \_\_beg, const \_CharT \* \_\_end ) const [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

**Parameters**

|        |                                              |
|--------|----------------------------------------------|
| __s    | The stream to write to.                      |
| __io   | Source of locale.                            |
| __fill | char_type to use for padding.                |
| __tm   | Struct tm with date and time info to format. |
| __beg  | Start of format string.                      |
| __end  | End of format string.                        |

**Returns**

Iterator after writing.

Definition at line 1143 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::narrow().

**4.978.2.3** template<typename \_CharT, typename \_Outlter > iter\_type std::time\_put< \_CharT, \_Outlter >::put ( iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const tm \* \_\_tm, char \_\_format, char \_\_mod = 0 ) const [inline], [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

**Parameters**

|          |                                              |
|----------|----------------------------------------------|
| __s      | The stream to write to.                      |
| __io     | Source of locale.                            |
| __fill   | char_type to use for padding.                |
| __tm     | Struct tm with date and time info to format. |
| __format | Format char.                                 |
| __mod    | Optional modifier char.                      |

**Returns**

Iterator after writing.

Definition at line 775 of file locale\_facets\_nonio.h.

References `std::time_put< _CharT, _OutIter >::do_put()`.

**4.978.3 Member Data Documentation**

4.978.3.1 `template<typename _CharT, typename _OutIter> locale::id std::time_put< _CharT, _OutIter >::id` `[static]`, `[inherited]`

Numpunct facet id.

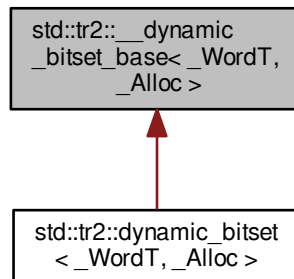
Definition at line 726 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

**4.979 std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc > Struct Template Reference**

Inheritance diagram for `std::tr2::__dynamic_bitset_base< _WordT, _Alloc >`:

**Public Types**

- `typedef _Alloc allocator_type`
- `typedef _WordT block_type`
- `typedef size_t size_type`

**Public Member Functions**

- `__dynamic_bitset_base` (`const allocator_type &__alloc=allocator_type()`)
- `__dynamic_bitset_base` (`__dynamic_bitset_base &&__b`)

- **\_\_dynamic\_bitset\_base** (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type())
- size\_t **M\_are\_all\_aux** () const
- void **M\_assign** (const \_\_dynamic\_bitset\_base &\_\_b)
- void **M\_clear** ()
- void **M\_do\_and** (const \_\_dynamic\_bitset\_base &\_\_x)
- void **M\_do\_append\_block** (block\_type \_\_block, size\_type \_\_pos)
- size\_t **M\_do\_count** () const
- void **M\_do\_dif** (const \_\_dynamic\_bitset\_base &\_\_x)
- size\_type **M\_do\_find\_first** (size\_t \_\_not\_found) const
- size\_type **M\_do\_find\_next** (size\_t \_\_prev, size\_t \_\_not\_found) const
- void **M\_do\_flip** ()
- void **M\_do\_left\_shift** (size\_t \_\_shift)
- void **M\_do\_or** (const \_\_dynamic\_bitset\_base &\_\_x)
- void **M\_do\_reset** ()
- void **M\_do\_right\_shift** (size\_t \_\_shift)
- void **M\_do\_set** ()
- unsigned long long **M\_do\_to\_ullong** () const
- unsigned long **M\_do\_to\_ulong** () const
- void **M\_do\_xor** (const \_\_dynamic\_bitset\_base &\_\_x)
- allocator\_type **M\_get\_allocator** () const
- block\_type & **M\_getword** (size\_type \_\_pos)
- block\_type **M\_getword** (size\_type \_\_pos) const
- block\_type & **M\_hiword** ()
- block\_type **M\_hiword** () const
- bool **M\_is\_any** () const
- bool **M\_is\_equal** (const \_\_dynamic\_bitset\_base &\_\_x) const
- bool **M\_is\_less** (const \_\_dynamic\_bitset\_base &\_\_x) const
- bool **M\_is\_proper\_subset\_of** (const \_\_dynamic\_bitset\_base &\_\_b) const
- bool **M\_is\_subset\_of** (const \_\_dynamic\_bitset\_base &\_\_b)
- void **M\_resize** (size\_t \_\_nbits, bool \_\_value)
- size\_type **M\_size** () const noexcept
- void **M\_swap** (\_\_dynamic\_bitset\_base &\_\_b)

#### Static Public Member Functions

- static block\_type **S\_maskbit** (size\_type \_\_pos) noexcept
- static size\_type **S\_whichbit** (size\_type \_\_pos) noexcept
- static size\_type **S\_whichbyte** (size\_type \_\_pos) noexcept
- static size\_type **S\_whichword** (size\_type \_\_pos) noexcept

#### Public Attributes

- std::vector< block\_type,  
allocator\_type > **M\_w**

#### Static Public Attributes

- static const size\_type **S\_bits\_per\_block**
- static const size\_type **npos**

## 4.979.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> struct std::tr2::__dynamic_bitset_
base< _WordT, _Alloc >
```

Base class, general case.

See documentation for `dynamic_bitset`.

Definition at line 80 of file `dynamic_bitset`.

## 4.979.2 Member Data Documentation

```
4.979.2.1 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w
```

0 is the least significant word.

Definition at line 93 of file `dynamic_bitset`.

The documentation for this struct was generated from the following file:

- [dynamic\\_bitset](#)

4.980 `std::tr2::__reflection_typelist<_First, _Rest...>` Struct Template Reference

## Public Types

- typedef [std::false\\_type](#) `empty`

## 4.980.1 Detailed Description

```
template<typename _First, typename... _Rest> struct std::tr2::__reflection_typelist<_First, _Rest...>
```

Partial specialization.

Definition at line 67 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

4.981 `std::tr2::__reflection_typelist<>` Struct Template Reference

## Public Types

- typedef [std::true\\_type](#) `empty`

## 4.981.1 Detailed Description

```
template<> struct std::tr2::__reflection_typelist<>
```

Specialization for an empty typelist.

Definition at line 60 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 4.982 `std::tr2::bases< _Tp >` Struct Template Reference

### Public Types

- typedef [\\_\\_reflection\\_typelist](#)  
`< __bases(_Tp)...> type`

### 4.982.1 Detailed Description

```
template<typename _Tp>struct std::tr2::bases< _Tp >
```

Sequence abstraction metafunctions for manipulating a typelist.

Enumerate all the base classes of a class. Form of a typelist.

Definition at line 88 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 4.983 `std::tr2::bool_set` Class Reference

### Public Member Functions

- constexpr [bool\\_set](#) ()
- constexpr [bool\\_set](#) (bool \_\_t)
- bool **contains** ([bool\\_set](#) \_\_b) const
- bool **equals** ([bool\\_set](#) \_\_b) const
- bool **is\_emptyset** () const
- bool **is\_indeterminate** () const
- bool **is\_singleton** () const
- **operator bool** () const

### Static Public Member Functions

- static [bool\\_set](#) **emptyset** ()
- static [bool\\_set](#) **indeterminate** ()

### Friends

- [bool\\_set](#) **operator!** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator&** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT,  
Traits > & **operator<<** ([std::basic\\_ostream](#)< CharT, Traits > &\_\_out, [bool\\_set](#) \_\_b)

- `bool_set operator== (bool_set __s, bool_set __t)`
- `template<typename CharT, typename Traits >  
std::basic_istream< CharT,  
Traits > & operator>> (std::basic_istream< CharT, Traits > &__in, bool_set &__b)`
- `bool_set operator^ (bool_set __s, bool_set __t)`
- `bool_set operator| (bool_set __s, bool_set __t)`

#### 4.983.1 Detailed Description

bool\_set

See N2136, Bool\_set: multi-valued logic by Hervnnimann, Guillaume Melquiond, Sylvain Pion.

The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

Definition at line 54 of file bool\_set.

#### 4.983.2 Constructor & Destructor Documentation

4.983.2.1 `constexpr std::tr2::bool_set::bool_set ( ) [inline]`

Default constructor.

Definition at line 59 of file bool\_set.

4.983.2.2 `constexpr std::tr2::bool_set::bool_set ( bool __f ) [inline]`

Constructor from bool.

Definition at line 62 of file bool\_set.

#### 4.983.3 Member Function Documentation

4.983.3.1 `bool std::tr2::bool_set::equals ( bool_set __b ) const [inline]`

Return true if states are equal.

Definition at line 69 of file bool\_set.

4.983.3.2 `bool std::tr2::bool_set::is_emptyset ( ) const [inline]`

Return true if this is empty.

Definition at line 73 of file bool\_set.

4.983.3.3 `bool std::tr2::bool_set::is_indeterminate ( ) const [inline]`

Return true if this is indeterminate.

Definition at line 77 of file bool\_set.

4.983.3.4 `bool std::tr2::bool_set::is_singleton ( ) const [inline]`

Return true if this is false or true (normal boolean).

Definition at line 81 of file bool\_set.

Referenced by operator `bool()`.

4.983.3.5 `std::tr2::bool_set::operator bool ( ) const` `[inline]`

Conversion to `bool`.

Definition at line 86 of file `bool_set`.

References `is_singleton()`.

The documentation for this class was generated from the following files:

- [bool\\_set](#)
- [bool\\_set.tcc](#)

## 4.984 `std::tr2::direct_bases<_Tp>` Struct Template Reference

Public Types

- typedef `__reflection_typelist<__direct_bases(_Tp)...>` **type**

### 4.984.1 Detailed Description

`template<typename _Tp> struct std::tr2::direct_bases<_Tp>`

Enumerate all the direct base classes of a class. Form of a `typelist`.

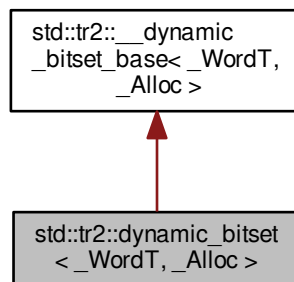
Definition at line 95 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 4.985 `std::tr2::dynamic_bitset<_WordT, _Alloc>` Class Template Reference

Inheritance diagram for `std::tr2::dynamic_bitset<_WordT, _Alloc>`:



## Classes

- class [reference](#)

## Public Types

- typedef [\\_\\_dynamic\\_bitset\\_base](#)  
    < \_WordT, \_Alloc > **\_Base**
- typedef \_Alloc **allocator\_type**
- typedef \_WordT **block\_type**
- typedef bool **const\_reference**
- typedef size\_t **size\_type**

## Public Member Functions

- [dynamic\\_bitset](#) (const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) ([initializer\\_list](#)< block\_type > \_\_il, const allocator\_type &\_\_alloc=allocator\_type())
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
    [dynamic\\_bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, typename [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_pos=0, typename [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'), const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (const char \* \_\_str, const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (const [dynamic\\_bitset](#) &\_\_b)
- [dynamic\\_bitset](#) ([dynamic\\_bitset](#) &&\_\_b)
- template<typename \_CharT, typename \_Traits >  
    void **\_M\_copy\_from\_ptr** (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT, \_CharT)
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
    void **\_M\_copy\_from\_string** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
    void **\_M\_copy\_to\_string** ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- bool [all](#) () const
- bool [any](#) () const
- void [append](#) (block\_type \_\_block)
- void [append](#) ([initializer\\_list](#)< block\_type > \_\_il)
- template<typename \_BlockInputIterator >  
    void [append](#) (\_BlockInputIterator \_\_first, \_BlockInputIterator \_\_last)
- void [clear](#) ()
- size\_type [count](#) () const noexcept
- bool [empty](#) () const noexcept
- size\_type [find\\_first](#) () const
- size\_type [find\\_next](#) (size\_t \_\_prev) const
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [flip](#) ()
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [flip](#) (size\_type \_\_pos)
- allocator\_type [get\\_allocator](#) () const
- bool **is\_proper\_subset\_of** (const [dynamic\\_bitset](#) &\_\_b) const
- bool **is\_subset\_of** (const [dynamic\\_bitset](#) &\_\_b) const

- constexpr size\_type [max\\_size](#) () noexcept
- bool [none](#) () const
- size\_type [num\\_blocks](#) () const noexcept
- [dynamic\\_bitset](#) & [operator=](#) (const [dynamic\\_bitset](#) & \_\_b)
- [dynamic\\_bitset](#) & [operator=](#) ([dynamic\\_bitset](#) && \_\_b)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > [operator~](#) () const
- void [push\\_back](#) (bool \_\_bit)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [reset](#) ()
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [reset](#) (size\_type \_\_pos)
- void [resize](#) (size\_type \_\_nbits, bool \_\_value=false)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [set](#) ()
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [set](#) (size\_type \_\_pos, bool \_\_val=true)
- size\_type [size](#) () const noexcept
- void [swap](#) ([dynamic\\_bitset](#) & \_\_b)
- bool [test](#) (size\_type \_\_pos) const
- template<typename \_CharT = char, typename \_Traits = std::char\_traits<\_CharT>, typename \_Alloc1 = std::allocator<\_CharT>>  
[std::basic\\_string](#)< \_CharT,  
 \_Traits, \_Alloc1 > [to\\_string](#) (\_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- unsigned long long [to\\_ullong](#) () const
- unsigned long [to\\_ulong](#) () const
  
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator&=](#) (const [dynamic\\_bitset](#)< \_WordT, \_Alloc > & \_\_rhs)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator&=](#) ([dynamic\\_bitset](#)< \_WordT, \_Alloc > && \_\_rhs)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator|=](#) (const [dynamic\\_bitset](#)< \_WordT, \_Alloc > & \_\_rhs)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator^=](#) (const [dynamic\\_bitset](#)< \_WordT, \_Alloc > & \_\_rhs)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator-=](#) (const [dynamic\\_bitset](#)< \_WordT, \_Alloc > & \_\_rhs)
  
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator<<=](#) (size\_type \_\_pos)
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > & [operator>>=](#) (size\_type \_\_pos)
  
- [reference operator\[\]](#) (size\_type \_\_pos)
- [const\\_reference operator\[\]](#) (size\_type \_\_pos) const
  
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > [operator<<](#) (size\_type \_\_pos) const
- [dynamic\\_bitset](#)< \_WordT, \_Alloc > [operator>>](#) (size\_type \_\_pos) const

#### Static Public Attributes

- static const size\_type **bits\_per\_block**
- static const size\_type **npos**

#### Private Member Functions

- size\_t **\_M\_are\_all\_aux** () const
- void **\_M\_assign** (const [\\_\\_dynamic\\_bitset\\_base](#) & \_\_b)
- void **\_M\_clear** ()
- void **\_M\_do\_and** (const [\\_\\_dynamic\\_bitset\\_base](#) & \_\_x)
- void **\_M\_do\_append\_block** (block\_type \_\_block, size\_type \_\_pos)
- size\_t **\_M\_do\_count** () const
- void **\_M\_do\_dif** (const [\\_\\_dynamic\\_bitset\\_base](#) & \_\_x)

- size\_type **\_M\_do\_find\_first** (size\_t \_\_not\_found) const
- size\_type **\_M\_do\_find\_next** (size\_t \_\_prev, size\_t \_\_not\_found) const
- void **\_M\_do\_flip** ()
- void **\_M\_do\_left\_shift** (size\_t \_\_shift)
- void **\_M\_do\_or** (const \_\_dynamic\_bitset\_base &\_\_x)
- void **\_M\_do\_reset** ()
- void **\_M\_do\_right\_shift** (size\_t \_\_shift)
- void **\_M\_do\_set** ()
- unsigned long long **\_M\_do\_to\_ullong** () const
- unsigned long **\_M\_do\_to\_ulong** () const
- void **\_M\_do\_xor** (const \_\_dynamic\_bitset\_base &\_\_x)
- allocator\_type **\_M\_get\_allocator** () const
- block\_type & **\_M\_getword** (size\_type \_\_pos)
- block\_type **\_M\_getword** (size\_type \_\_pos) const
- block\_type & **\_M\_hiword** ()
- block\_type **\_M\_hiword** () const
- bool **\_M\_is\_any** () const
- bool **\_M\_is\_equal** (const \_\_dynamic\_bitset\_base &\_\_x) const
- bool **\_M\_is\_less** (const \_\_dynamic\_bitset\_base &\_\_x) const
- bool **\_M\_is\_proper\_subset\_of** (const \_\_dynamic\_bitset\_base &\_\_b) const
- bool **\_M\_is\_subset\_of** (const \_\_dynamic\_bitset\_base &\_\_b)
- void **\_M\_resize** (size\_t \_\_nbits, bool \_\_value)
- size\_type **\_M\_size** () const noexcept
- void **\_M\_swap** (\_\_dynamic\_bitset\_base &\_\_b)

#### Static Private Member Functions

- static block\_type **\_S\_maskbit** (size\_type \_\_pos) noexcept
- static size\_type **\_S\_whichbit** (size\_type \_\_pos) noexcept
- static size\_type **\_S\_whichbyte** (size\_type \_\_pos) noexcept
- static size\_type **\_S\_whichword** (size\_type \_\_pos) noexcept

#### Private Attributes

- [std::vector](#)< block\_type,  
allocator\_type > **\_M\_w**

#### Static Private Attributes

- static const size\_type **\_S\_bits\_per\_block**

#### Friends

- class **reference**

## 4.985.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> class std::tr2::dynamic_bitset< _WordT, _Alloc >
```

The `dynamic_bitset` class represents a sequence of bits.

(Note that `dynamic_bitset` does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let *B* be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage. *B* - *NbB* bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `dynamic_bitset` as "a simple array of bits," be aware that your mental picture is reversed: a `dynamic_bitset` behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and the bit at index *Nb*-1 in the "most significant / left-hand" position. Thus, unlike other containers, a `dynamic_bitset`'s index "counts from right to left," to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
 long a = 'a';
 dynamic_bitset b(a);

 cout << "b('a') is " << b << endl;

 ostringstream s;
 s << b;
 string str = s.str();
 cout << "index 3 in the string is " << str[3] << " but\n"
 << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch33s02.html> for a description of extensions.

Most of the actual code isn't contained in `dynamic_bitset<>` itself, but in the base class `__dynamic_bitset_base`. The base class works with whole words, not with individual bits. This allows us to specialize `__dynamic_bitset_base` for the important special case where the `dynamic_bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `__dynamic_bitset_base` is a vector, and is indexed as such. This is carefully encapsulated.

Definition at line 569 of file `dynamic_bitset`.

## 4.985.2 Constructor &amp; Destructor Documentation

```
4.985.2.1 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const allocator_type & __alloc =
allocator_type()) [inline],[explicit]
```

All bits set to zero.

Definition at line 721 of file `dynamic_bitset`.

```
4.985.2.2 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (size_type __nbits, unsigned long long __val =
0ULL, const allocator_type & __alloc = allocator_type()) [inline],[explicit]
```

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 727 of file dynamic\_bitset.

```
4.985.2.3 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> template<typename
_CharT, typename _Traits, typename _Alloc1> std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset
(const std::basic_string< _CharT, _Traits, _Alloc1 > & __str, typename basic_string< _CharT, _Traits,
_Alloc1 >::size_type __pos = 0, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n =
std::basic_string<_CharT, _Traits, _Alloc1>::npos, _CharT __zero = _CharT('0'), _CharT
__one = _CharT('1'), const allocator_type & __alloc = allocator_type()) [inline],[explicit]
```

Use a subset of a string.

#### Parameters

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <code>__str</code> | A string of '0' and '1' characters.                        |
| <code>__pos</code> | Index of the first character in <code>__str</code> to use. |
| <code>__n</code>   | The number of characters to copy.                          |

#### Exceptions

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::out_of_range</code>     | If <code>__pos</code> is bigger the size of <code>__str</code> .   |
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |

Definition at line 749 of file dynamic\_bitset.

References `std::tr2::dynamic_bitset< _WordT, _Alloc >::resize()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
4.985.2.4 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const char * __str, const allocator_type & __alloc =
allocator_type()) [inline],[explicit]
```

Construct from a string.

#### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__str</code> | A string of '0' and '1' characters. |
|--------------------|-------------------------------------|

#### Exceptions

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |
|------------------------------------|--------------------------------------------------------------------|

Definition at line 777 of file dynamic\_bitset.

References `std::tr2::dynamic_bitset< _WordT, _Alloc >::resize()`.

```
4.985.2.5 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const dynamic_bitset< _WordT, _Alloc > & __b
) [inline]
```

Copy constructor.

Definition at line 793 of file dynamic\_bitset.

```
4.985.2.6 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset(dynamic_bitset< _WordT, _Alloc > && __b)
[inline]
```

Move constructor.

Definition at line 800 of file dynamic\_bitset.

#### 4.985.3 Member Function Documentation

```
4.985.3.1 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool
std::tr2::dynamic_bitset< _WordT, _Alloc >::all() const [inline]
```

Tests whether all the bits are on.

##### Returns

True if all the bits are set.

Definition at line 1187 of file dynamic\_bitset.

```
4.985.3.2 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool
std::tr2::dynamic_bitset< _WordT, _Alloc >::any() const [inline]
```

Tests whether any of the bits are on.

##### Returns

True if at least one bit is set.

Definition at line 1195 of file dynamic\_bitset.

```
4.985.3.3 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void
std::tr2::dynamic_bitset< _WordT, _Alloc >::append(block_type __block) [inline]
```

Append a block.

Definition at line 881 of file dynamic\_bitset.

Referenced by std::tr2::dynamic\_bitset<\_WordT, \_Alloc >::append().

```
4.985.3.4 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> template<typename
_BlockInputIterator > void std::tr2::dynamic_bitset< _WordT, _Alloc >::append(_BlockInputIterator __first,
_BlockInputIterator __last) [inline]
```

Append an iterator range of blocks.

Definition at line 899 of file dynamic\_bitset.

References std::tr2::dynamic\_bitset<\_WordT, \_Alloc >::append().

```
4.985.3.5 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void
std::tr2::dynamic_bitset< _WordT, _Alloc >::clear() [inline]
```

Clear the bitset.

Definition at line 859 of file dynamic\_bitset.

4.985.3.6 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type  
std::tr2::dynamic_bitset<_WordT, _Alloc >::count ( ) const [inline], [noexcept]`

Returns the number of bits which are set.

Definition at line 1143 of file dynamic\_bitset.

4.985.3.7 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool  
std::tr2::dynamic_bitset<_WordT, _Alloc >::empty ( ) const [inline], [noexcept]`

Returns true if the dynamic\_bitset is empty.

Definition at line 1158 of file dynamic\_bitset.

4.985.3.8 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type  
std::tr2::dynamic_bitset<_WordT, _Alloc >::find_first ( ) const [inline]`

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See Also

[find\\_next](#)

Definition at line 1223 of file dynamic\_bitset.

4.985.3.9 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type  
std::tr2::dynamic_bitset<_WordT, _Alloc >::find_next ( size_t __prev ) const [inline]`

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__prev</code> | Where to start searching. |
|---------------------|---------------------------|

See Also

[find\\_first](#)

Definition at line 1233 of file dynamic\_bitset.

4.985.3.10 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc >::flip ( ) [inline]`

Toggles every bit to its opposite value.

Definition at line 1038 of file dynamic\_bitset.

Referenced by `std::tr2::dynamic_bitset<_WordT, _Alloc >::operator~()`.

```
4.985.3.11 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
 dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc >::flip (size_type __pos)
 [inline]
```

Toggles a given bit to its opposite value.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

#### Exceptions

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Definition at line 1051 of file `dynamic_bitset`.

```
4.985.3.12 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> allocator_type
 std::tr2::dynamic_bitset<_WordT, _Alloc >::get_allocator () const [inline]
```

Return the allocator for the bitset.

Definition at line 841 of file `dynamic_bitset`.

```
4.985.3.13 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> constexpr size_type
 std::tr2::dynamic_bitset<_WordT, _Alloc >::max_size () [inline], [noexcept]
```

Returns the maximum size of a `dynamic_bitset` object having the same type as `*this`. The real answer is `max() * bits_per_block` but is likely to overflow.

Definition at line 1165 of file `dynamic_bitset`.

References `std::numeric_limits<_Tp >::max()`.

```
4.985.3.14 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool
 std::tr2::dynamic_bitset<_WordT, _Alloc >::none () const [inline]
```

Tests whether any of the bits are on.

#### Returns

True if none of the bits are set.

Definition at line 1203 of file `dynamic_bitset`.

```
4.985.3.15 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type
 std::tr2::dynamic_bitset<_WordT, _Alloc >::num_blocks () const [inline], [noexcept]
```

Returns the total number of blocks.

Definition at line 1153 of file `dynamic_bitset`.

```
4.985.3.16 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
 dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator&= (const
 dynamic_bitset<_WordT, _Alloc > &__rhs) [inline]
```

Operations on `dynamic_bitsets`.

## Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

Definition at line 914 of file `dynamic_bitset`.

```
4.985.3.17 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator&= (
dynamic_bitset<_WordT, _Alloc > && __rhs) [inline]
```

Operations on `dynamic_bitsets`.

## Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

Definition at line 921 of file `dynamic_bitset`.

```
4.985.3.18 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator= (const
dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on `dynamic_bitsets`.

## Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

Definition at line 942 of file `dynamic_bitset`.

```
4.985.3.19 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset<_WordT, _Alloc> std::tr2::dynamic_bitset<_WordT, _Alloc >::operator<< (size_type __pos)
const [inline]
```

Self-explanatory.

Definition at line 1209 of file `dynamic_bitset`.

```
4.985.3.20 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator<<= (size_type
__pos) [inline]
```

Operations on `dynamic_bitsets`.

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

Definition at line 957 of file `dynamic_bitset`.

4.985.3.21 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc>::operator= ( const dynamic_bitset<_WordT, _Alloc> & _b ) [inline]`

Assignment.

Definition at line 818 of file `dynamic_bitset`.

4.985.3.22 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc>::operator= ( dynamic_bitset<_WordT, _Alloc> && _b ) [inline]`

Move assignment.

Definition at line 831 of file `dynamic_bitset`.

References `std::tr2::dynamic_bitset<_WordT, _Alloc>::swap()`.

4.985.3.23 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc> std::tr2::dynamic_bitset<_WordT, _Alloc>::operator>> ( size_type __pos ) const [inline]`

Self-explanatory.

Definition at line 1213 of file `dynamic_bitset`.

4.985.3.24 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc>::operator>=> ( size_type __pos ) [inline]`

Operations on dynamic\_bitsets.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

Definition at line 970 of file `dynamic_bitset`.

4.985.3.25 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> reference std::tr2::dynamic_bitset<_WordT, _Alloc>::operator[] ( size_type __pos ) [inline]`

Array-indexing support.

#### Parameters

|                    |                                              |
|--------------------|----------------------------------------------|
| <code>__pos</code> | Index into the <code>dynamic_bitset</code> . |
|--------------------|----------------------------------------------|

#### Returns

A bool for a 'const `dynamic_bitset`'. For non-const bitsets, an instance of the reference proxy class.

#### Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 1073 of file `dynamic_bitset`.

4.985.3.26 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> const_reference  
std::tr2::dynamic_bitset<_WordT, _Alloc>::operator[]( size_type __pos ) const [inline]`

Array-indexing support.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index into the dynamic_bitset. |
|--------------------|--------------------------------|

#### Returns

A bool for a 'const dynamic\_bitset'. For non-const bitsets, an instance of the reference proxy class.

#### Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 1077 of file dynamic\_bitset.

4.985.3.27 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc>::operator^= ( const  
dynamic_bitset<_WordT, _Alloc> & __rhs ) [inline]`

Operations on dynamic\_bitsets.

#### Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

Definition at line 935 of file dynamic\_bitset.

4.985.3.28 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset<_WordT, _Alloc>::operator|= ( const  
dynamic_bitset<_WordT, _Alloc> & __rhs ) [inline]`

Operations on dynamic\_bitsets.

#### Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

Definition at line 928 of file dynamic\_bitset.

4.985.3.29 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc> std::tr2::dynamic_bitset<_WordT, _Alloc>::operator~ ( ) const  
[inline]`

See the no-argument flip().

Definition at line 1060 of file dynamic\_bitset.

References std::tr2::dynamic\_bitset<\_WordT, \_Alloc>::flip().

4.985.3.30 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void  
std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back( bool __bit ) [inline]`

Push a bit onto the high end of the bitset.

Definition at line 869 of file `dynamic_bitset`.

References `std::tr2::dynamic_bitset< _WordT, _Alloc >::size()`.

4.985.3.31 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset ( ) [inline]`

Sets every bit to false.

Definition at line 1013 of file `dynamic_bitset`.

4.985.3.32 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset ( size_type __pos )  
[inline]`

Sets a given bit to false.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

#### Exceptions

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Same as writing `set ( __pos, false )`.

Definition at line 1027 of file `dynamic_bitset`.

4.985.3.33 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void  
std::tr2::dynamic_bitset< _WordT, _Alloc >::resize ( size_type __nbits, bool __value = false ) [inline]`

Resize the bitset.

Definition at line 848 of file `dynamic_bitset`.

Referenced by `std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset()`, and `std::tr2::operator>>()`.

4.985.3.34 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::set ( ) [inline]`

Sets every bit to true.

Definition at line 988 of file `dynamic_bitset`.

4.985.3.35 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>  
dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::set ( size_type __pos, bool  
__val = true ) [inline]`

Sets a given bit to a particular value.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__pos</code> | The index of the bit.                   |
| <code>__val</code> | Either true or false, defaults to true. |

## Exceptions

|                          |                                                      |
|--------------------------|------------------------------------------------------|
| <i>std::out_of_range</i> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------|------------------------------------------------------|

Definition at line 1002 of file `dynamic_bitset`.

4.985.3.36 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type  
std::tr2::dynamic_bitset< _WordT, _Alloc >::size ( ) const [inline], [noexcept]`

Returns the total number of bits.

Definition at line 1148 of file `dynamic_bitset`.

Referenced by `std::tr2::operator>>()`, and `std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back()`.

4.985.3.37 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void  
std::tr2::dynamic_bitset< _WordT, _Alloc >::swap ( dynamic_bitset< _WordT, _Alloc > & __b ) [inline]`

Swap with another bitset.

Definition at line 808 of file `dynamic_bitset`.

Referenced by `std::tr2::dynamic_bitset< _WordT, _Alloc >::operator=()`.

4.985.3.38 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool  
std::tr2::dynamic_bitset< _WordT, _Alloc >::test ( size_type __pos ) const [inline]`

Tests the value of a bit.

## Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__pos</code> | The index of a bit. |
|--------------------|---------------------|

## Returns

The value at `__pos`.

## Exceptions

|                          |                                                      |
|--------------------------|------------------------------------------------------|
| <i>std::out_of_range</i> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------|------------------------------------------------------|

Definition at line 1175 of file `dynamic_bitset`.

4.985.3.39 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> template<typename  
_CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 = std::allocator<_CharT>>  
std::basic_string<_CharT, _Traits, _Alloc1> std::tr2::dynamic_bitset< _WordT, _Alloc >::to_string ( _CharT  
__zero = _CharT('0'), _CharT __one = _CharT('1') ) const [inline]`

Returns a character interpretation of the `dynamic_bitset`.

## Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1113 of file `dynamic_bitset`.

4.985.3.40 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> unsigned long long  
std::tr2::dynamic_bitset<_WordT, _Alloc>::to_ulong ( ) const [inline]`

Returns a numerical interpretation of the `dynamic_bitset`.

#### Returns

The integral equivalent of the bits.

#### Exceptions

|                                  |                                                                                 |
|----------------------------------|---------------------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an <code>unsigned long</code> . |
|----------------------------------|---------------------------------------------------------------------------------|

Definition at line 1098 of file `dynamic_bitset`.

4.985.3.41 `template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> unsigned long  
std::tr2::dynamic_bitset<_WordT, _Alloc>::to_ulong ( ) const [inline]`

Returns a numerical interpretation of the `dynamic_bitset`.

#### Returns

The integral equivalent of the bits.

#### Exceptions

|                                  |                                                                                 |
|----------------------------------|---------------------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an <code>unsigned long</code> . |
|----------------------------------|---------------------------------------------------------------------------------|

Definition at line 1088 of file `dynamic_bitset`.

The documentation for this class was generated from the following file:

- [dynamic\\_bitset](#)

## 4.986 `std::tr2::dynamic_bitset<_WordT, _Alloc>::reference` Class Reference

### Public Member Functions

- **reference** ([dynamic\\_bitset](#) &\_\_b, size\_type \_\_pos)
- [reference](#) & **flip** ()
- **operator bool** () const
- [reference](#) & **operator=** (bool \_\_x)
- [reference](#) & **operator=** (const [reference](#) &\_\_j)
- bool **operator~** () const

### Friends

- class **dynamic\_bitset**

### 4.986.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)

Definition at line 654 of file dynamic\_bitset.

The documentation for this class was generated from the following file:

- [dynamic\\_bitset](#)

## 4.987 std::try\_to\_lock\_t Struct Reference

### 4.987.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

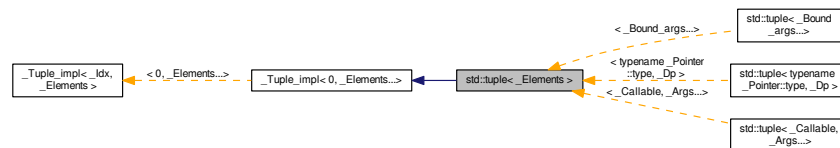
Definition at line 392 of file mutex.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 4.988 std::tuple< \_Elements > Class Template Reference

Inheritance diagram for std::tuple< \_Elements >:



### Public Member Functions

- constexpr **tuple** (const \_Elements &... \_\_elements)
- template<typename... \_UElements, typename = typename enable\_if<\_\_and<is\_convertible<\_UElements, \_Elements>...>::value>::type>
   
constexpr **tuple** (\_UElements &&... \_\_elements)
- constexpr **tuple** (const [tuple](#) &)=default
- constexpr **tuple** ([tuple](#) &&)=default
- template<typename... \_UElements, typename = typename enable\_if<\_\_and<is\_convertible<const \_UElements&, \_Elements>...>::value>::type>
   
constexpr **tuple** (const [tuple](#)< \_UElements...> & \_\_in)

- `template<typename... _UElements, typename = typename enable_if<__and<is_convertible<_UElements, _Elements>...>::value>::type>`  
`constexpr tuple (tuple<_UElements...> &&__in)`
- `template<typename _Alloc >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, const _Elements &...__elements)`
- `template<typename _Alloc, typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&...__elements)`
- `template<typename _Alloc >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)`
- `template<typename _Alloc >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc, typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple<_UElements...> &__in)`
- `template<typename _Alloc, typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, tuple<_UElements...> &&__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in) noexcept(is_nothrow_move_assignable<_Inherited>)`
- `template<typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>`  
`tuple & operator= (const tuple<_UElements...> &__in)`
- `template<typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>`  
`tuple & operator= (tuple<_UElements...> &&__in)`
- `void swap (tuple &__in) noexcept(noexcept(__in._M_swap(__in)))`

#### 4.988.1 Detailed Description

`template<typename... _Elements>class std::tuple<_Elements>`

Primary class template, tuple.

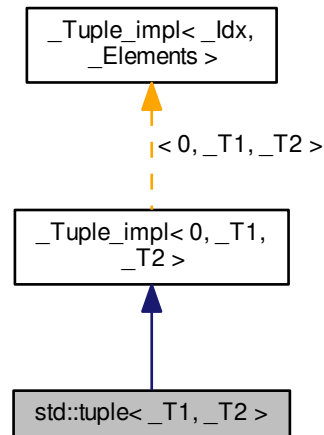
Definition at line 388 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 4.989 std::tuple&lt; \_T1, \_T2 &gt; Class Template Reference

Inheritance diagram for std::tuple< \_T1, \_T2 >:



## Public Member Functions

- constexpr **tuple** (const \_T1 &\_\_a1, const \_T2 &\_\_a2)
- template<typename \_U1 , typename \_U2 , typename = typename enable\_if<\_\_and<is\_convertible<\_U1, \_T1>, is\_convertible<\_U2, \_T2>>::value>::type>  
constexpr **tuple** (\_U1 &&\_\_a1, \_U2 &&\_\_a2)
- constexpr **tuple** (const **tuple** &)=default
- constexpr **tuple** (**tuple** &&)=default
- template<typename \_U1 , typename \_U2 , typename = typename enable\_if<\_\_and<is\_convertible<const \_U1&, \_T1>, is\_convertible<const \_U2&, \_T2>>::value>::type>  
constexpr **tuple** (const **tuple** < \_U1, \_U2 > &\_\_in)
- template<typename \_U1 , typename \_U2 , typename = typename enable\_if<\_\_and<is\_convertible<\_U1, \_T1>, is\_convertible<\_U2, \_T2>>::value>::type>  
constexpr **tuple** (**tuple** < \_U1, \_U2 > &&\_\_in)
- template<typename \_U1 , typename \_U2 , typename = typename enable\_if<\_\_and<is\_convertible<const \_U1&, \_T1>, is\_convertible<const \_U2&, \_T2>>::value>::type>  
constexpr **tuple** (const **pair** < \_U1, \_U2 > &\_\_in)
- template<typename \_U1 , typename \_U2 , typename = typename enable\_if<\_\_and<is\_convertible<\_U1, \_T1>, is\_convertible<\_U2, \_T2>>::value>::type>  
constexpr **tuple** (**pair** < \_U1, \_U2 > &&\_\_in)
- template<typename \_Alloc >  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a)
- template<typename \_Alloc >  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, const \_T1 &\_\_a1, const \_T2 &\_\_a2)
- template<typename \_Alloc , typename \_U1 , typename \_U2 >  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, \_U1 &&\_\_a1, \_U2 &&\_\_a2)

- `template<typename _Alloc >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)`
- `template<typename _Alloc >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2 >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)`
- `template<typename _Alloc, typename _U1, typename _U2 >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2 >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)`
- `template<typename _Alloc, typename _U1, typename _U2 >`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in) noexcept(is_nothrow_move_assignable< _Inherited >)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (const tuple< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (const pair< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (pair< _U1, _U2 > &&__in)`
- `void swap (tuple &__in) noexcept(noexcept(__in._M_swap(__in)))`

#### 4.989.1 Detailed Description

`template<typename _T1, typename _T2>class std::tuple< _T1, _T2 >`

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.

Definition at line 521 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 4.990 `std::tuple_element< 0, tuple< _Head, _Tail...> >` Struct Template Reference

### Public Types

- `typedef _Head type`

#### 4.990.1 Detailed Description

`template<typename _Head, typename... _Tail>struct std::tuple_element< 0, tuple< _Head, _Tail...> >`

Basis case for tuple\_element: The first element is the one we're seeking.

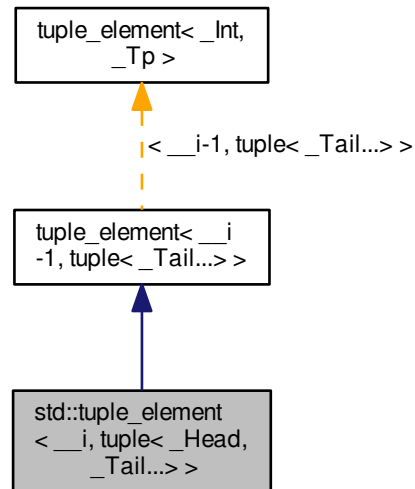
Definition at line 687 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

4.991 `std::tuple_element< __i, tuple< _Head, _Tail...> >` Struct Template Reference

Inheritance diagram for `std::tuple_element< __i, tuple< _Head, _Tail...> >`:



## 4.991.1 Detailed Description

```
template<std::size_t __i, typename _Head, typename... _Tail>struct std::tuple_element< __i, tuple< _Head, _Tail...> >
```

Recursive case for `tuple_element`: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

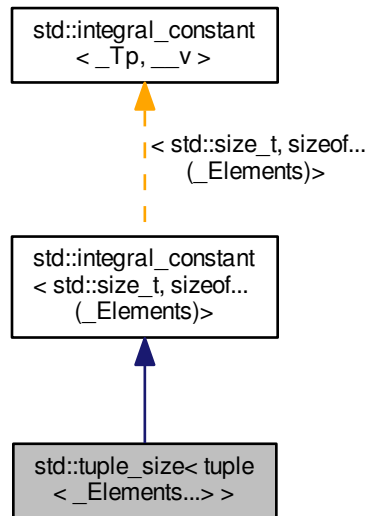
Definition at line 680 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.992 std::tuple\_size&lt; tuple&lt; \_Elements...&gt; &gt; Struct Template Reference

Inheritance diagram for std::tuple\_size< tuple< \_Elements...> >:



## Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `std::size_t` **value**

## 4.992.1 Detailed Description

```
template<typename... _Elements> struct std::tuple_size< tuple< _Elements...> >
```

```
class tuple_size
```

Definition at line 737 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.993 std::type\_index Struct Reference

### Public Member Functions

- **type\_index** (const [type\\_info](#) &\_\_rhs) noexcept
- **size\_t hash\_code** () const noexcept
- const char \* **name** () const
- bool **operator!=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator==** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>=** (const [type\\_index](#) &\_\_rhs) const noexcept

### 4.993.1 Detailed Description

Class `type_index`

The class `type_index` provides a simple wrapper for `type_info` which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Definition at line 52 of file `typeidindex`.

The documentation for this struct was generated from the following file:

- [typeidindex](#)

## 4.994 std::type\_info Class Reference

Inherited by `__cxxabiv1::__array_type_info`, `__cxxabiv1::__class_type_info`, `__cxxabiv1::__enum_type_info`, `__cxxabiv1::__function_type_info`, `__cxxabiv1::__fundamental_type_info`, and `__cxxabiv1::__pbase_type_info`.

### Public Member Functions

- virtual `~type_info` ()
- virtual bool **do\_catch** (const [type\\_info](#) \*\_\_thr\_type, void \*\*\_\_thr\_obj, unsigned \_\_outer) const
- virtual bool **do\_upcast** (const `__cxxabiv1::__class_type_info` \*\_\_target, void \*\*\_\_obj\_ptr) const
- virtual bool **is\_function\_p** () const
- virtual bool **is\_pointer\_p** () const
- bool **before** (const [type\\_info](#) &\_\_arg) const noexcept
- **size\_t hash\_code** () const noexcept
- const char \* **name** () const noexcept
- bool **operator!=** (const [type\\_info](#) &\_\_arg) const noexcept
- bool **operator==** (const [type\\_info](#) &\_\_arg) const noexcept

### Protected Member Functions

- **type\_info** (const char \*\_\_n)

### Protected Attributes

- const char \* **\_\_name**

#### 4.994.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

Definition at line 88 of file `typeinfo`.

#### 4.994.2 Constructor & Destructor Documentation

##### 4.994.2.1 virtual std::type\_info::~type\_info ( ) [virtual]

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated `type_info` structures in the new-abi.

#### 4.994.3 Member Function Documentation

##### 4.994.3.1 const char\* std::type\_info::name ( ) const [inline], [noexcept]

Returns an *implementation-defined* byte string; this is not portable between compilers!

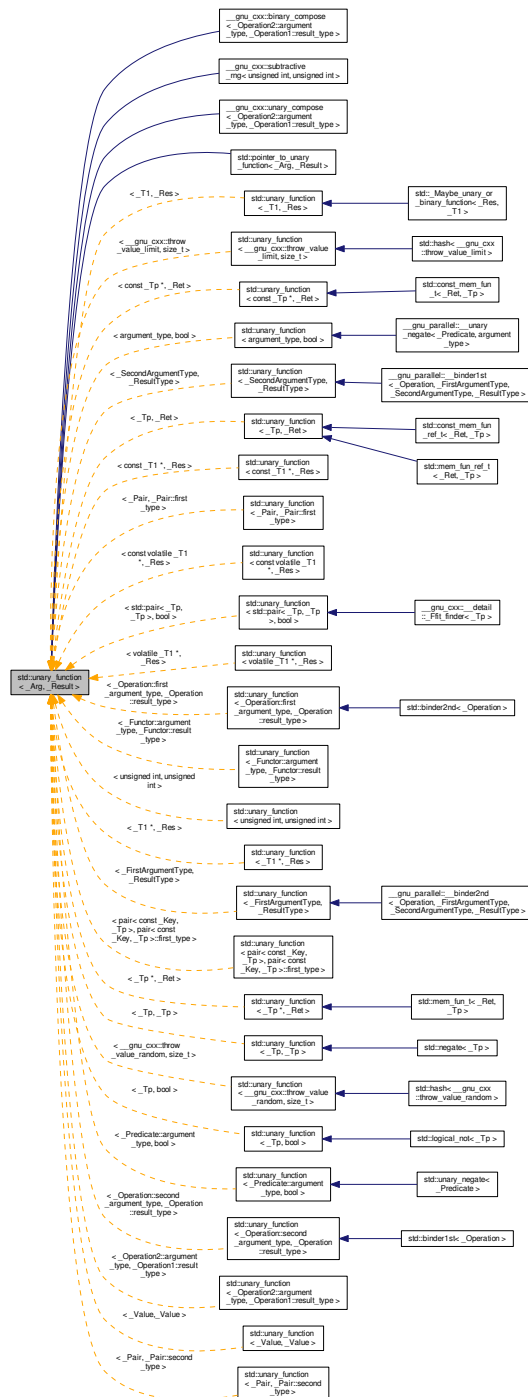
Definition at line 99 of file `typeinfo`.

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 4.995 std::unary\_function&lt; \_Arg, \_Result &gt; Struct Template Reference

Inheritance diagram for std::unary\_function< \_Arg, \_Result >:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### 4.995.1 Detailed Description

`template<typename _Arg, typename _Result> struct std::unary_function< _Arg, _Result >`

This is one of the [functor base classes](#).

Definition at line 101 of file `stl_function.h`.

### 4.995.2 Member Typedef Documentation

**4.995.2.1** `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type`

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

**4.995.2.2** `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type`

`result_type` is the return type

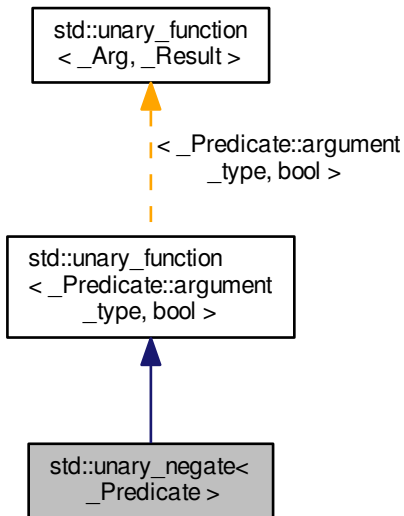
Definition at line 107 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.996 std::unary\_negate&lt; \_Predicate &gt; Class Template Reference

Inheritance diagram for std::unary\_negate< \_Predicate >:



## Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- **unary\_negate** (const `_Predicate` &\_\_x)
- `bool operator()` (const typename `_Predicate::argument_type` &\_\_x) const

## Protected Attributes

- `_Predicate` **\_M\_pred**

## 4.996.1 Detailed Description

```
template<typename _Predicate>class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 351 of file `stl_function.h`.

## 4.996.2 Member Typedef Documentation

4.996.2.1 `typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type, bool >::argument_type`  
[*inherited*]

`argument_type` is the type of the argument

Definition at line 104 of file `stl_function.h`.

4.996.2.2 `typedef bool std::unary_function< _Predicate::argument_type, bool >::result_type` [*inherited*]

`result_type` is the return type

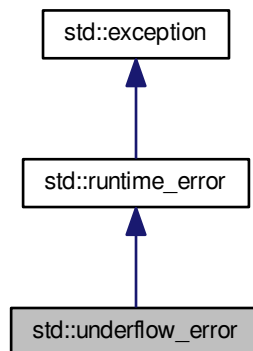
Definition at line 107 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

4.997 `std::underflow_error` Class Reference

Inheritance diagram for `std::underflow_error`:



## Public Member Functions

- **`underflow_error`** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const noexcept

## 4.997.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 146 of file `stdexcept`.

## 4.997.2 Member Function Documentation

4.997.2.1 `virtual const char* std::runtime_error::what ( ) const` `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

4.998 `std::underlying_type<_Tp>` Struct Template Reference

## Public Member Functions

- `typedef __underlying_type (_Tp) type`

## 4.998.1 Detailed Description

```
template<typename _Tp>struct std::underlying_type<_Tp>
```

The underlying type of an enum.

Definition at line 1855 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.999 `std::uniform_int_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- `typedef _IntType result_type`

## Public Member Functions

- [uniform\\_int\\_distribution](#) (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType>::max()`)
- [uniform\\_int\\_distribution](#) (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`

- `result_type a () const`
- `result_type b () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `bool operator== (const uniform_int_distribution &__d1, const uniform_int_distribution &__d2)`

#### 4.999.1 Detailed Description

`template<typename _IntType = int> class std::uniform_int_distribution< _IntType >`

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Definition at line 1666 of file random.h.

#### 4.999.2 Member Typedef Documentation

4.999.2.1 `template<typename _IntType = int> typedef _IntType std::uniform_int_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 1669 of file random.h.

#### 4.999.3 Constructor & Destructor Documentation

4.999.3.1 `template<typename _IntType = int> std::uniform_int_distribution< _IntType >::uniform_int_distribution ( _IntType __a = 0, _IntType __b = std::numeric_limits<_IntType>::max() ) [inline], [explicit]`

Constructs a uniform distribution object.

Definition at line 1709 of file random.h.

#### 4.999.4 Member Function Documentation

4.999.4.1 `template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::max ( ) const [inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1761 of file random.h.

4.999.4.2 `template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::min ( ) const`  
`[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1754 of file random.h.

4.999.4.3 `template<typename _IntType = int> template<typename UniformRandomNumberGenerator > result_type`  
`std::uniform_int_distribution< _IntType >::operator() ( UniformRandomNumberGenerator & _urng )`  
`[inline]`

Generating functions.

Definition at line 1769 of file random.h.

References `std::uniform_int_distribution< _IntType >::operator()()`.

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`.

4.999.4.4 `template<typename _IntType = int> param_type std::uniform_int_distribution< _IntType >::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 1739 of file random.h.

Referenced by `std::operator>>()`.

4.999.4.5 `template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::param ( const param_type`  
`& _param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 1747 of file random.h.

4.999.4.6 `template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1725 of file random.h.

#### 4.999.5 Friends And Related Function Documentation

4.999.5.1 `template<typename _IntType = int> bool operator== ( const uniform_int_distribution< _IntType > & _d1, const`  
`uniform_int_distribution< _IntType > & _d2 ) [friend]`

Return true if two uniform integer distributions have the same parameters.

Definition at line 1804 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.1000 `std::uniform_int_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef [uniform\\_int\\_distribution](#)  
`< _IntType >` **distribution\_type**

### Public Member Functions

- **param\_type** (`_IntType __a=0, _IntType __b=std::numeric_limits< _IntType >::max()`)
- **result\_type** **a** () const
- **result\_type** **b** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.1000.1 Detailed Description

template<typename `_IntType` = int>struct `std::uniform_int_distribution< _IntType >::param_type`

Parameter type.

Definition at line 1675 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.1001 `std::uniform_real_distribution< _RealType >` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` **result\_type**

### Public Member Functions

- [uniform\\_real\\_distribution](#) (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- **uniform\_real\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)

- `template<typename _UniformRandomNumberGenerator >`  
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `result_type a () const`
- `result_type b () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `bool operator== (const uniform_real_distribution &__d1, const uniform_real_distribution &__d2)`

#### 4.1001.1 Detailed Description

`template<typename _RealType = double>class std::uniform_real_distribution< _RealType >`

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1867 of file random.h.

#### 4.1001.2 Member Typedef Documentation

4.1001.2.1 `template<typename _RealType = double> typedef _RealType std::uniform_real_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 1870 of file random.h.

#### 4.1001.3 Constructor & Destructor Documentation

4.1001.3.1 `template<typename _RealType = double> std::uniform_real_distribution< _RealType >::uniform_real_distribution ( _RealType __a = _RealType (0), _RealType __b = _RealType (1) )`  
`[inline], [explicit]`

Constructs a uniform\_real\_distribution object.

#### Parameters

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__a</code> | [IN] The lower bound of the distribution. |
| <code>__b</code> | [IN] The upper bound of the distribution. |

Definition at line 1913 of file `random.h`.

#### 4.1001.4 Member Function Documentation

4.1001.4.1 `template<typename _RealType = double> result_type std::uniform_real_distribution<_RealType>::max ( )`  
`const [inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1965 of file `random.h`.

4.1001.4.2 `template<typename _RealType = double> result_type std::uniform_real_distribution<_RealType>::min ( )`  
`const [inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1958 of file `random.h`.

4.1001.4.3 `template<typename _RealType = double> template<typename UniformRandomNumberGenerator> result_type`  
`std::uniform_real_distribution<_RealType>::operator() ( UniformRandomNumberGenerator & __urng )`  
`[inline]`

Generating functions.

Definition at line 1973 of file `random.h`.

References `std::uniform_real_distribution<_RealType>::operator()()`.

Referenced by `std::uniform_real_distribution<_RealType>::operator()()`.

4.1001.4.4 `template<typename _RealType = double> param_type std::uniform_real_distribution<_RealType>::param (`  
`) const [inline]`

Returns the parameter set of the distribution.

Definition at line 1943 of file `random.h`.

Referenced by `std::operator>>()`.

4.1001.4.5 `template<typename _RealType = double> void std::uniform_real_distribution<_RealType>::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 1951 of file `random.h`.

4.1001.4.6 `template<typename _RealType = double> void std::uniform_real_distribution<_RealType>::reset ( )`  
`[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1929 of file `random.h`.

## 4.1001.5 Friends And Related Function Documentation

4.1001.5.1 `template<typename _RealType = double> bool operator==( const uniform_real_distribution<_RealType> & __d1, const uniform_real_distribution<_RealType> & __d2 )` [`friend`]

Return true if two uniform real distributions have the same parameters.

Definition at line 2013 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.1002 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef [uniform\\_real\\_distribution<\\_RealType>](#) **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- [result\\_type a](#) () const
- [result\\_type b](#) () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.1002.1 Detailed Description

`template<typename _RealType = double> struct std::uniform_real_distribution<_RealType>::param_type`

Parameter type.

Definition at line 1876 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.1003 `std::unique_lock<_Mutex>` Class Template Reference

## Public Types

- typedef `_Mutex` **mutex\_type**

## Public Member Functions

- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#))
- template<typename \_Clock, typename \_Duration >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Rep, typename \_Period >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- **unique\_lock** (const [unique\\_lock](#) &)=delete
- **unique\_lock** ([unique\\_lock](#) &&\_\_u) noexcept
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- [unique\\_lock](#) & **operator=** (const [unique\\_lock](#) &)=delete
- [unique\\_lock](#) & **operator=** ([unique\\_lock](#) &&\_\_u) noexcept
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** ([unique\\_lock](#) &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

## 4.1003.1 Detailed Description

```
template<typename _Mutex>class std::unique_lock< _Mutex >
```

[unique\\_lock](#)

Definition at line 429 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

## 4.1004 std::unique\_ptr&lt; \_Tp, \_Dp &gt; Class Template Reference

## Public Types

- typedef \_Dp **deleter\_type**
- typedef \_Tp **element\_type**
- typedef \_Pointer::type **pointer**

## Public Member Functions

- **unique\_ptr** (pointer \_\_p) noexcept
- **unique\_ptr** (pointer \_\_p, typename [conditional](#)< [is\\_reference](#)< deleter\_type >::value, deleter\_type, const deleter\_type & >::type \_\_d) noexcept
- **unique\_ptr** (pointer \_\_p, typename [remove\\_reference](#)< deleter\_type >::type &&\_\_d) noexcept
- constexpr **unique\_ptr** (nullptr\_t) noexcept
- **unique\_ptr** ([unique\\_ptr](#) &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep, typename = \_Require< is\_convertible<typename unique\_ptr<\_Up, \_Ep>::pointer, pointer>, \_not\_<is\_array<\_Up>>, typename conditional<is\_reference<\_Dp>::value, is\_same<\_Ep, \_Dp>, is\_convertible<\_Ep, \_Dp>>::type>> **unique\_ptr** ([unique\\_ptr](#)< \_Up, \_Ep > &&\_\_u) noexcept
- **unique\_ptr** (const [unique\\_ptr](#) &)=delete
- template<typename \_Up, typename > **unique\_ptr** ([auto\\_ptr](#)< \_Up > &&\_\_u) noexcept
- pointer **get** () const noexcept
- deleter\_type & **get\_deleter** () noexcept
- const deleter\_type & **get\_deleter** () const noexcept
- **operator bool** () const noexcept
- [add\\_lvalue\\_reference](#)  
< element\_type >::type **operator\*** () const
- pointer **operator->** () const noexcept
- [unique\\_ptr](#) & **operator=** ([unique\\_ptr](#) &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep > [enable\\_if](#)< \_\_and\_< [is\\_convertible](#)< typename [unique\\_ptr](#)< \_Up, \_Ep >::pointer, pointer >, \_\_not\_< [is\\_array](#)< \_Up > > >::value, [unique\\_ptr](#) & >::type **operator=** ([unique\\_ptr](#)< \_Up, \_Ep > &&\_\_u) noexcept
- [unique\\_ptr](#) & **operator=** (nullptr\_t) noexcept
- [unique\\_ptr](#) & **operator=** (const [unique\\_ptr](#) &)=delete
- pointer **release** () noexcept
- void **reset** (pointer \_\_p=pointer()) noexcept
- void **swap** ([unique\\_ptr](#) &\_\_u) noexcept

## 4.1004.1 Detailed Description

```
template<typename _Tp, typename _Dp = default_delete<_Tp>> class std::unique_ptr< _Tp, _Dp >
```

20.7.1.2 unique\_ptr for single objects.

Definition at line 109 of file unique\_ptr.h.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 4.1005 std::unique\_ptr&lt; \_Tp[], \_Dp &gt; Class Template Reference

## Public Types

- typedef \_Dp **deleter\_type**
- typedef \_Tp **element\_type**
- typedef \_Pointer::type **pointer**

## Public Member Functions

- **unique\_ptr** (pointer \_\_p) noexcept
- template<typename \_Up, typename = \_Require<is\_pointer<pointer>, is\_convertible<\_Up\*, pointer>, \_\_is\_derived\_Tp<\_Up>>>  
**unique\_ptr** (\_Up \*\_\_p)=delete
- **unique\_ptr** (pointer \_\_p, typename conditional< is\_reference< deleter\_type >::value, deleter\_type, const deleter\_type & >::type \_\_d) noexcept
- **unique\_ptr** (pointer \_\_p, typename remove\_reference< deleter\_type >::type &&\_\_d) noexcept
- **unique\_ptr** (**unique\_ptr** &&\_\_u) noexcept
- constexpr **unique\_ptr** (nullptr\_t) noexcept
- template<typename \_Up, typename \_Ep, typename = \_Require<\_\_safe\_conversion<\_Up, \_Ep>, typename conditional<is\_reference<\_Dp>::value, is\_same<\_Ep, \_Dp>, is\_convertible<\_Ep, \_Dp>>::type >>  
**unique\_ptr** (**unique\_ptr**< \_Up, \_Ep > &&\_\_u) noexcept
- **unique\_ptr** (const **unique\_ptr** &)=delete
- template<typename \_Up, typename = \_Require<is\_pointer<pointer>, is\_convertible<\_Up\*, pointer>, \_\_is\_derived\_Tp<\_Up>>>  
**unique\_ptr** (\_Up \*, typename conditional< is\_reference< deleter\_type >::value, deleter\_type, const deleter\_type & >::type)=delete
- template<typename \_Up, typename = \_Require<is\_pointer<pointer>, is\_convertible<\_Up\*, pointer>, \_\_is\_derived\_Tp<\_Up>>>  
**unique\_ptr** (\_Up \*, typename remove\_reference< deleter\_type >::type &&)=delete
- pointer **get** () const noexcept
- deleter\_type & **get\_deleter** () noexcept
- const deleter\_type & **get\_deleter** () const noexcept
- **operator bool** () const noexcept
- **unique\_ptr** & **operator=** (**unique\_ptr** &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep >  
enable\_if< \_\_safe\_conversion  
< \_Up, \_Ep >::value,  
**unique\_ptr** & >::type **operator=** (**unique\_ptr**< \_Up, \_Ep > &&\_\_u) noexcept
- **unique\_ptr** & **operator=** (nullptr\_t) noexcept
- **unique\_ptr** & **operator=** (const **unique\_ptr** &)=delete
- **std::add\_lvalue\_reference**  
< element\_type >::type **operator[]** (size\_t \_\_i) const
- pointer **release** () noexcept
- void **reset** () noexcept
- void **reset** (pointer \_\_p) noexcept
- template<typename \_Up, typename = \_Require<is\_pointer<pointer>, is\_convertible<\_Up\*, pointer>, \_\_is\_derived\_Tp<\_Up>>>  
void **reset** (\_Up \*)=delete
- void **swap** (**unique\_ptr** &&\_\_u) noexcept

## 4.1005.1 Detailed Description

```
template<typename _Tp, typename _Dp>class std::unique_ptr< _Tp[], _Dp >
```

20.7.1.3 unique\_ptr for array objects with a runtime length

Definition at line 282 of file unique\_ptr.h.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 4.1006 std::unordered\_map&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inherits std::\_\_allow\_copy\_cons< bool >.

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::mapped\_type [mapped\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef allocator\_type::pointer [pointer](#)
- typedef allocator\_type::const\_pointer [const\\_pointer](#)
- typedef allocator\_type::reference [reference](#)
- typedef allocator\_type::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_map](#) (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[unordered\\_map](#) (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [unordered\\_map](#) (const unordered\_map &)=default
- [unordered\\_map](#) (unordered\_map &&)=default
- [unordered\\_map](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [iterator](#) begin () noexcept

- [local\\_iterator begin](#) ([size\\_type](#) \_\_n)
  - [size\\_type bucket](#) (const [key\\_type](#) &\_\_key) const
  - [size\\_type bucket\\_count](#) () const noexcept
  - [size\\_type bucket\\_size](#) ([size\\_type](#) \_\_n) const
  - void [clear](#) () noexcept
  - [size\\_type count](#) (const [key\\_type](#) &\_\_x) const
  - template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&...\_\_args)
  - template<typename... \_Args>  
[iterator](#) [emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&...\_\_args)
  - bool [empty](#) () const noexcept
  - [iterator end](#) () noexcept
  - [local\\_iterator end](#) ([size\\_type](#) \_\_n)
  - [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
  - [iterator erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
  - [allocator\\_type get\\_allocator](#) () const noexcept
  - [hasher hash\\_function](#) () const
  - template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void [insert](#) (initializer\_list< [value\\_type](#) > \_\_l)
  - [key\\_equal key\\_eq](#) () const
  - float [load\\_factor](#) () const noexcept
  - [size\\_type max\\_bucket\\_count](#) () const noexcept
  - float [max\\_load\\_factor](#) () const noexcept
  - void [max\\_load\\_factor](#) (float \_\_z)
  - [size\\_type max\\_size](#) () const noexcept
  - [unordered\\_map & operator=](#) (const [unordered\\_map](#) &)=default
  - [unordered\\_map & operator=](#) ([unordered\\_map](#) &&)=default
  - [unordered\\_map & operator=](#) (initializer\_list< [value\\_type](#) > \_\_l)
  - void [rehash](#) ([size\\_type](#) \_\_n)
  - void [reserve](#) ([size\\_type](#) \_\_n)
  - [size\\_type size](#) () const noexcept
  - void [swap](#) ([unordered\\_map](#) &\_\_x)
- 
- [const\\_iterator begin](#) () const noexcept
  - [const\\_iterator cbegin](#) () const noexcept
- 
- [const\\_iterator end](#) () const noexcept
  - [const\\_iterator cend](#) () const noexcept
- 
- [std::pair](#)< [iterator](#), bool > [insert](#) (const [value\\_type](#) &\_\_x)
  - template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
[std::pair](#)< [iterator](#), bool > [insert](#) (\_Pair &&\_\_x)
- 
- [iterator insert](#) (const [iterator](#) \_\_hint, const [value\\_type](#) &\_\_x)
  - template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
[iterator insert](#) (const [iterator](#) \_\_hint, \_Pair &&\_\_x)
- 
- [iterator erase](#) (const [iterator](#) \_\_position)
  - [iterator erase](#) ([iterator](#) \_\_it)

- [iterator find](#) (const [key\\_type](#) &\_\_x)
- [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
- [std::pair< iterator, iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x)
- [std::pair< const\\_iterator, const\\_iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x) const
- [mapped\\_type & operator\[\]](#) (const [key\\_type](#) &\_\_k)
- [mapped\\_type & operator\[\]](#) ([key\\_type](#) &&\_\_k)
- [mapped\\_type & at](#) (const [key\\_type](#) &\_\_k)
- const [mapped\\_type & at](#) (const [key\\_type](#) &\_\_k) const
- [const\\_local\\_iterator begin](#) ([size\\_type](#) \_\_n) const
- [const\\_local\\_iterator cbegin](#) ([size\\_type](#) \_\_n) const
- [const\\_local\\_iterator end](#) ([size\\_type](#) \_\_n) const
- [const\\_local\\_iterator cend](#) ([size\\_type](#) \_\_n) const

#### Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 >  
bool operator== (const unordered\_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered\_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)`

#### 4.1006.1 Detailed Description

`template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

#### Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                              |
| <code>_Tp</code>    | Type of mapped objects.                                                           |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

Definition at line 97 of file `unordered_map.h`.

#### 4.1006.2 Member Typedef Documentation

4.1006.2.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::allocator_type`

Public typedefs.

Definition at line 111 of file unordered\_map.h.

4.1006.2.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 121 of file unordered\_map.h.

4.1006.2.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 123 of file unordered\_map.h.

4.1006.2.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::const_pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 117 of file unordered\_map.h.

4.1006.2.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::const_reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 119 of file unordered\_map.h.

4.1006.2.6 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::difference_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::difference_type`

Iterator-related typedefs.

Definition at line 125 of file unordered\_map.h.

4.1006.2.7 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::hasher`

Public typedefs.

Definition at line 109 of file unordered\_map.h.

4.1006.2.8 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::iterator`

Iterator-related typedefs.

Definition at line 120 of file unordered\_map.h.

4.1006.2.9 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::key_equal`

Public typedefs.

Definition at line 110 of file unordered\_map.h.

4.1006.2.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::key_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::key_type`

Public typedefs.

Definition at line 106 of file unordered\_map.h.

4.1006.2.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::local_iterator`

Iterator-related typedefs.

Definition at line 122 of file unordered\_map.h.

4.1006.2.12 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::mapped_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::mapped_type`

Public typedefs.

Definition at line 108 of file unordered\_map.h.

4.1006.2.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::pointer`

Iterator-related typedefs.

Definition at line 116 of file unordered\_map.h.

4.1006.2.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef allocator_type::reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::reference`

Iterator-related typedefs.

Definition at line 118 of file unordered\_map.h.

4.1006.2.15 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::size_type`

Iterator-related typedefs.

Definition at line 124 of file unordered\_map.h.

4.1006.2.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::value_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::value_type`

Public typedefs.

Definition at line 107 of file unordered\_map.h.

#### 4.1006.3 Constructor & Destructor Documentation

4.1006.3.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map( size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline],[explicit]`

Default constructor creates no elements.

##### Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__n</code>   | Initial number of buckets. |
| <code>__hf</code>  | A hash functor.            |
| <code>__eqf</code> | A key equality functor.    |
| <code>__a</code>   | An allocator object.       |

Definition at line 138 of file unordered\_map.h.

4.1006.3.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map( _InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline]`

Builds an unordered\_map from a range.

##### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_map consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is distance(`__first`,`__last`)).

Definition at line 159 of file unordered\_map.h.

4.1006.3.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map( const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]`

Copy constructor.

4.1006.3.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]`

Move constructor.

4.1006.3.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map( initializer_list<value_type> __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline]`

Builds an unordered\_map from an initializer\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_map consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 184 of file unordered\_map.h.

#### 4.1006.4 Member Function Documentation

4.1006.4.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::at( const key_type & __k ) [inline]`

Access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the unordered\_map.

#### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

Definition at line 612 of file unordered\_map.h.

4.1006.4.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::at( const key_type & __k ) const [inline]`

Access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the unordered\_map.

#### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

Definition at line 616 of file unordered\_map.h.

4.1006.4.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::begin( ) [inline], [noexcept]`

Returns a read/write iterator that points to the first element in the unordered\_map.

Definition at line 248 of file unordered\_map.h.

4.1006.4.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::begin( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_map.

Definition at line 257 of file unordered\_map.h.

4.1006.4.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::begin( size_type __n ) [inline]`

Returns a read/write iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read/write local iterator.

Definition at line 657 of file unordered\_map.h.

4.1006.4.6 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::begin( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read-only local iterator.

Definition at line 668 of file unordered\_map.h.

```
4.1006.4.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
 >::bucket_count () const [inline], [noexcept]
```

Returns the number of buckets of the unordered\_map.

Definition at line 624 of file unordered\_map.h.

```
4.1006.4.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
 _Alloc>::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_map.

Definition at line 261 of file unordered\_map.h.

```
4.1006.4.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
 _Pred, _Alloc>::cbegin (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read-only local iterator.

Definition at line 672 of file unordered\_map.h.

```
4.1006.4.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
 _Alloc>::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

Definition at line 283 of file unordered\_map.h.

```
4.1006.4.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
 _Pred, _Alloc>::cend (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read-only local iterator.

Definition at line 698 of file unordered\_map.h.

```
4.1006.4.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 497 of file unordered\_map.h.

```
4.1006.4.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::count (const key_type & __x) const [inline]
```

Finds the number of elements.

## Parameters

|                  |               |
|------------------|---------------|
| <code>__x</code> | Key to count. |
|------------------|---------------|

## Returns

Number of elements with specified key.

This function only makes sense for unordered\_multimap; for unordered\_map the result will either be 0 (not present) or 1 (present).

Definition at line 560 of file unordered\_map.h.

```
4.1006.4.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> std::pair<iterator, bool>
 std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::emplace (_Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_map.

## Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

Definition at line 310 of file unordered\_map.h.

```
4.1006.4.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (const_iterator __pos, _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_map.

#### Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

#### Returns

An iterator that points to the element with key of the std::pair built from `__args` (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 340 of file unordered\_map.h.

```
4.1006.4.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> bool std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::empty () const [inline], [noexcept]
```

Returns true if the unordered\_map is empty.

Definition at line 228 of file unordered\_map.h.

```
4.1006.4.17 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered\_map.

Definition at line 270 of file unordered\_map.h.

```
4.1006.4.18 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

Definition at line 279 of file unordered\_map.h.

```
4.1006.4.19 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc >::end (size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read/write local iterator.

Definition at line 683 of file unordered\_map.h.

```
4.1006.4.20 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map< _Key, _Tp, _Hash,
_Pred, _Alloc >::end (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read-only local iterator.

Definition at line 694 of file unordered\_map.h.

```
4.1006.4.21 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::unordered_map< _Key, _Tp,
_Hash, _Pred, _Alloc >::equal_range (const key_type & __x) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered\_multimap.

Definition at line 573 of file unordered\_map.h.

```
4.1006.4.22 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & __x) const
[inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered\_multimap.

Definition at line 577 of file unordered\_map.h.

```
4.1006.4.23 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
 >::erase(const_iterator __position) [inline]
```

Erases an element from an unordered\_map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 447 of file unordered\_map.h.

```
4.1006.4.24 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
 >::erase(iterator __it) [inline]
```

Erases an element from an unordered\_map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 452 of file unordered\_map.h.

```
4.1006.4.25 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
 >::erase(const key_type & __x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_map`. For an `unordered_map` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 469 of file `unordered_map.h`.

```
4.1006.4.26 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
 >::erase(const_iterator __first, const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_map`.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_map`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 487 of file `unordered_map.h`.

```
4.1006.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
 >::find(const key_type & __x) [inline]
```

Tries to locate an element in an `unordered_map`.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 542 of file unordered\_map.h.

```
4.1006.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
 _Alloc>::find (const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_map.

**Parameters**

|     |                    |
|-----|--------------------|
| __x | Key to be located. |
|-----|--------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 546 of file unordered\_map.h.

```
4.1006.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred,
 _Alloc>::get_allocator () const [inline],[noexcept]
```

Returns the allocator object with which the unordered\_map was constructed.

Definition at line 221 of file unordered\_map.h.

```
4.1006.4.30 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
 >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_map was constructed.

Definition at line 518 of file unordered\_map.h.

```
4.1006.4.31 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::unordered_map<_Key, _Tp,
 _Hash, _Pred, _Alloc>::insert (const value_type & __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

**Parameters**

|     |                                                                      |
|-----|----------------------------------------------------------------------|
| __x | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|-----|----------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

Definition at line 362 of file unordered\_map.h.

```
4.1006.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
 _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
 std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair<iterator, bool>
 std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::insert(_Pair && __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

#### Parameters

|     |                                                                      |
|-----|----------------------------------------------------------------------|
| __x | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|-----|----------------------------------------------------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

Definition at line 369 of file unordered\_map.h.

```
4.1006.4.33 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
 >::insert(const_iterator __hint, const value_type & __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

#### Parameters

|        |                                                                            |
|--------|----------------------------------------------------------------------------|
| __hint | An iterator that serves as a hint as to where the pair should be inserted. |
| __x    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

#### Returns

An iterator that points to the element with key of \_\_x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 396 of file unordered\_map.h.

4.1006.4.34 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::insert ( const_iterator __hint, _Pair && __x ) [inline]`

Attempts to insert a std::pair into the unordered\_map.

#### Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 403 of file unordered\_map.h.

4.1006.4.35 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::insert ( _InputIterator __first, _InputIterator __last ) [inline]`

A template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 418 of file unordered\_map.h.

4.1006.4.36 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::insert ( initializer_list<value_type> & __l ) [inline]`

Attempts to insert a list of elements into the unordered\_map.

#### Parameters

|                  |                                                                 |
|------------------|-----------------------------------------------------------------|
| <code>__l</code> | A std::initializer_list<value_type> of elements to be inserted. |
|------------------|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 429 of file unordered\_map.h.

4.1006.4.37 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::key_eq ( ) const [inline]`

Returns the key comparison object with which the unordered\_map was constructed.

Definition at line 524 of file unordered\_map.h.

4.1006.4.38 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::load_factor ( ) const [inline], [noexcept]`

Returns the average number of elements per bucket.

Definition at line 706 of file unordered\_map.h.

4.1006.4.39 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_bucket_count ( ) const [inline], [noexcept]`

Returns the maximum number of buckets of the unordered\_map.

Definition at line 629 of file unordered\_map.h.

4.1006.4.40 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_load_factor ( ) const [inline], [noexcept]`

Returns a positive number that the unordered\_map tries to keep the load factor less than or equal to.

Definition at line 712 of file unordered\_map.h.

4.1006.4.41 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_load_factor ( float __z ) [inline]`

Change the unordered\_map maximum load factor.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

Definition at line 720 of file unordered\_map.h.

4.1006.4.42 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::max_size ( ) const [inline], [noexcept]`

Returns the maximum size of the unordered\_map.

Definition at line 238 of file unordered\_map.h.

4.1006.4.43 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator= ( const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]`

Copy assignment operator.

4.1006.4.44 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator=( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]`

Move assignment operator.

4.1006.4.45 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator=( initializer_list<value_type> &_l ) [inline]`

Unordered\_map list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_map and that the resulting unordered\_map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 212 of file unordered\_map.h.

4.1006.4.46 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[]( const key_type &_k ) [inline]`

Subscript ( `[]` ) access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 595 of file unordered\_map.h.

4.1006.4.47 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[]( key_type && _k ) [inline]`

Subscript ( `[]` ) access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] )operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 599 of file unordered\_map.h.

**4.1006.4.48** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::rehash ( size_type __n ) [inline]`

May rehash the unordered\_map.

**Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_map maximum load factor.

Definition at line 731 of file unordered\_map.h.

**4.1006.4.49** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::reserve ( size_type __n ) [inline]`

Prepare the unordered\_map for a specified number of elements.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as rehash(ceil(n / max\_load\_factor())).

Definition at line 742 of file unordered\_map.h.

**4.1006.4.50** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::size ( ) const [inline],[noexcept]`

Returns the size of the unordered\_map.

Definition at line 233 of file unordered\_map.h.

**4.1006.4.51** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::swap ( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & __x ) [inline]`

Swaps data with another unordered\_map.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__x</code> | An unordered_map of the same element and allocator types. |
|------------------|-----------------------------------------------------------|

This exchanges the elements between two unordered\_map in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 510 of file unordered\_map.h.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 4.1007 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inherits std::\_\_allow\_copy\_cons< bool >.

### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::mapped\_type [mapped\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef allocator\_type::pointer [pointer](#)
- typedef allocator\_type::const\_pointer [const\\_pointer](#)
- typedef allocator\_type::reference [reference](#)
- typedef allocator\_type::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

### Public Member Functions

- [unordered\\_multimap](#) ([size\\_type](#) \_\_n=10, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator >  
[unordered\\_multimap](#) (\_InputIterator \_\_f, \_InputIterator \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multimap](#) (const [unordered\\_multimap](#) &)=default
- [unordered\\_multimap](#) ([unordered\\_multimap](#) &&)=default
- [unordered\\_multimap](#) (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [iterator](#) [begin](#) () noexcept
- [local\\_iterator](#) [begin](#) ([size\\_type](#) \_\_n)
- [size\\_type](#) [bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) [bucket\\_count](#) () const noexcept
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_n) const
- void [clear](#) () noexcept

- [size\\_type count](#) (const [key\\_type](#) &\_\_x) const
- [template](#)<typename... \_Args>  
[iterator emplace](#) (\_Args &&...\_\_args)
- [template](#)<typename... \_Args>  
[iterator emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&...\_\_args)
- [bool empty](#) () const noexcept
- [iterator end](#) () noexcept
- [local\\_iterator end](#) ([size\\_type](#) \_\_n)
- [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
- [iterator erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [allocator\\_type get\\_allocator](#) () const noexcept
- [hasher hash\\_function](#) () const
- [template](#)<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (initializer\_list< [value\\_type](#) > \_\_l)
- [key\\_equal key\\_eq](#) () const
- [float load\\_factor](#) () const noexcept
- [size\\_type max\\_bucket\\_count](#) () const noexcept
- [float max\\_load\\_factor](#) () const noexcept
- void [max\\_load\\_factor](#) (float \_\_z)
- [size\\_type max\\_size](#) () const noexcept
- [unordered\\_multimap & operator=](#) (const [unordered\\_multimap](#) &)=default
- [unordered\\_multimap & operator=](#) ([unordered\\_multimap](#) &&)=default
- [unordered\\_multimap & operator=](#) (initializer\_list< [value\\_type](#) > \_\_l)
- void [rehash](#) ([size\\_type](#) \_\_n)
- void [reserve](#) ([size\\_type](#) \_\_n)
- [size\\_type size](#) () const noexcept
- void [swap](#) ([unordered\\_multimap](#) &\_\_x)
  
- [const\\_iterator begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
  
- [const\\_iterator end](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
  
- [iterator insert](#) (const [value\\_type](#) &\_\_x)
- [template](#)<typename \_Pair , typename = typename std::enable\_if<std::is\_constructible<[value\\_type](#), \_Pair&&>::value>::type>  
[iterator insert](#) (\_Pair &&\_\_x)
  
- [iterator insert](#) (const [iterator](#) \_\_hint, const [value\\_type](#) &\_\_x)
- [template](#)<typename \_Pair , typename = typename std::enable\_if<std::is\_constructible<[value\\_type](#), \_Pair&&>::value>::type>  
[iterator insert](#) (const [iterator](#) \_\_hint, \_Pair &&\_\_x)
  
- [iterator erase](#) (const [iterator](#) \_\_position)
- [iterator erase](#) ([iterator](#) \_\_it)
  
- [iterator find](#) (const [key\\_type](#) &\_\_x)
- [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
  
- [std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const [key\\_type](#) &\_\_x)

- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `const_local_iterator begin` (size\_type \_\_n) const
- `const_local_iterator cbegin` (size\_type \_\_n) const
- `const_local_iterator end` (size\_type \_\_n) const
- `const_local_iterator cend` (size\_type \_\_n) const

#### Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (const `unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

#### 4.1007.1 Detailed Description

`template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

#### Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                              |
| <code>_Tp</code>    | Type of mapped objects.                                                           |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

Definition at line 778 of file `unordered_map.h`.

#### 4.1007.2 Member Typedef Documentation

**4.1007.2.1** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::allocator_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type`

Public typedefs.

Definition at line 792 of file `unordered_map.h`.

**4.1007.2.2** `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 802 of file unordered\_map.h.

```
4.1007.2.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
 class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::const_local_iterator
 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

Definition at line 804 of file unordered\_map.h.

```
4.1007.2.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::const_pointer std::unordered_multimap<
 _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 798 of file unordered\_map.h.

```
4.1007.2.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
 class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::const_reference
 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference
```

Iterator-related typedefs.

Definition at line 800 of file unordered\_map.h.

```
4.1007.2.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::difference_type std::unordered_multimap<
 _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type
```

Iterator-related typedefs.

Definition at line 806 of file unordered\_map.h.

```
4.1007.2.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp,
 _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 790 of file unordered\_map.h.

```
4.1007.2.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp,
 _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 801 of file unordered\_map.h.

```
4.1007.2.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_equal std::unordered_multimap< _Key,
 _Tp, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 791 of file unordered\_map.h.

4.1007.2.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type`

Public typedefs.

Definition at line 787 of file unordered\_map.h.

4.1007.2.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator`

Iterator-related typedefs.

Definition at line 803 of file unordered\_map.h.

4.1007.2.12 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::mapped_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type`

Public typedefs.

Definition at line 789 of file unordered\_map.h.

4.1007.2.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer`

Iterator-related typedefs.

Definition at line 797 of file unordered\_map.h.

4.1007.2.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 799 of file unordered\_map.h.

4.1007.2.15 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type`

Iterator-related typedefs.

Definition at line 805 of file unordered\_map.h.

4.1007.2.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 788 of file unordered\_map.h.

#### 4.1007.3 Constructor & Destructor Documentation

```
4.1007.3.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::
unordered_multimap (size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf =
key_equal(), const allocator_type & __a = allocator_type()) [inline],[explicit]
```

Default constructor creates no elements.

#### Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__n</code>   | Initial number of buckets. |
| <code>__hf</code>  | A hash functor.            |
| <code>__eqf</code> | A key equality functor.    |
| <code>__a</code>   | An allocator object.       |

Definition at line 819 of file unordered\_map.h.

```
4.1007.3.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_multimap (_InputIterator __f, _InputIterator __l, size_type __n = 0,
const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a =
allocator_type()) [inline]
```

Builds an unordered\_multimap from a range.

#### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is `distance(__first,__last)`).

Definition at line 840 of file unordered\_map.h.

```
4.1007.3.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
>::unordered_multimap (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &) [default]
```

Copy constructor.

```
4.1007.3.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
>::unordered_multimap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &&) [default]
```

Move constructor.

```
4.1007.3.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_multimap (initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf =
hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type())
[inline]
```

Builds an unordered\_multimap from an initializer\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_multimap consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 865 of file unordered\_map.h.

#### 4.1007.4 Member Function Documentation

```
4.1007.4.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered\_multimap.

Definition at line 929 of file unordered\_map.h.

```
4.1007.4.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc>::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 938 of file unordered\_map.h.

```
4.1007.4.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc>::begin (size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read/write local iterator.

Definition at line 1277 of file unordered\_map.h.

```
4.1007.4.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_multimap< _Key, _Tp,
_Hash, _Pred, _Alloc >::begin (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1288 of file unordered\_map.h.

```
4.1007.4.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::bucket_count () const [inline], [noexcept]
```

Returns the number of buckets of the unordered\_multimap.

Definition at line 1244 of file unordered\_map.h.

```
4.1007.4.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 942 of file unordered\_map.h.

```
4.1007.4.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_multimap< _Key, _Tp,
_Hash, _Pred, _Alloc >::cbegin (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1292 of file unordered\_map.h.

```
4.1007.4.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap< _Key, _Tp, _Hash,
 _Pred, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

Definition at line 964 of file unordered\_map.h.

```
4.1007.4.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap< _Key, _Tp,
 _Hash, _Pred, _Alloc >::cend (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1318 of file unordered\_map.h.

```
4.1007.4.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
 >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1161 of file unordered\_map.h.

```
4.1007.4.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc >::count (const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__x</code> | Key to count. |
|------------------|---------------|

**Returns**

Number of elements with specified key.

Definition at line 1221 of file unordered\_map.h.

```
4.1007.4.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
 class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator
 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (_Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the unordered\_multimap.

Insertion requires amortized constant time.

Definition at line 987 of file unordered\_map.h.

```
4.1007.4.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
 class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator
 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (const_iterator __pos, _Args &&...
 __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the std::pair built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1013 of file unordered\_map.h.

```
4.1007.4.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
 >::empty () const [inline], [noexcept]
```

Returns true if the unordered\_multimap is empty.

Definition at line 909 of file unordered\_map.h.

```
4.1007.4.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered\_multimap.

Definition at line 951 of file unordered\_map.h.

```
4.1007.4.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

Definition at line 960 of file unordered\_map.h.

```
4.1007.4.17 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc >::end (size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read/write local iterator.

Definition at line 1303 of file unordered\_map.h.

```
4.1007.4.18 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap<_Key, _Tp,
_Hash, _Pred, _Alloc >::end (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read-only local iterator.

Definition at line 1314 of file unordered\_map.h.

```
4.1007.4.19 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & __x) [inline]
```

Finds a subsequence matching given key.

## Parameters

|     |                    |
|-----|--------------------|
| __x | Key to be located. |
|-----|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1232 of file unordered\_map.h.

```
4.1007.4.20 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
 _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator>
 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & __x) const
 [inline]
```

Finds a subsequence matching given key.

## Parameters

|     |                    |
|-----|--------------------|
| __x | Key to be located. |
|-----|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1236 of file unordered\_map.h.

```
4.1007.4.21 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc >::erase (const_iterator __position) [inline]
```

Erases an element from an unordered\_multimap.

## Parameters

|            |                                                   |
|------------|---------------------------------------------------|
| __position | An iterator pointing to the element to be erased. |
|------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following \_\_position prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1111 of file unordered\_map.h.

```
4.1007.4.22 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc >::erase (iterator __it) [inline]
```

Erases an element from an unordered\_multimap.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1116 of file `unordered_map.h`.

```
4.1007.4.23 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc >::erase (const key_type & _x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>__x</code> | Key of elements to be erased. |
|------------------|-------------------------------|

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1132 of file `unordered_map.h`.

```
4.1007.4.24 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
 std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc >::erase (const_iterator __first, const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1151 of file `unordered_map.h`.

4.1007.4.25 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find ( const key_type & __x ) [inline]`

Tries to locate an element in an unordered\_multimap.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1207 of file unordered\_map.h.

4.1007.4.26 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find ( const key_type & __x ) const [inline]`

Tries to locate an element in an unordered\_multimap.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1211 of file unordered\_map.h.

4.1007.4.27 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator ( ) const [inline], [noexcept]`

Returns the allocator object with which the unordered\_multimap was constructed.

Definition at line 902 of file unordered\_map.h.

4.1007.4.28 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]`

Returns the hash functor object with which the unordered\_multimap was constructed.

Definition at line 1183 of file unordered\_map.h.

4.1007.4.29 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert( const value_type & __x ) [inline]`

Inserts a std::pair into the unordered\_multimap.

#### Parameters

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

#### Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1027 of file unordered\_map.h.

4.1007.4.30 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert( _Pair && __x ) [inline]`

Inserts a std::pair into the unordered\_multimap.

#### Parameters

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

#### Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1034 of file unordered\_map.h.

4.1007.4.31 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert( const_iterator __hint, const value_type & __x ) [inline]`

Inserts a std::pair into the unordered\_multimap.

#### Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

#### Returns

An iterator that points to the element with key of \_\_x (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1059 of file unordered\_map.h.

```
4.1007.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert(const_iterator __hint, _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

#### Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1066 of file unordered\_map.h.

```
4.1007.4.33 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename InputIterator > void
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert(InputIterator __first, InputIterator __last)
[inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 1081 of file unordered\_map.h.

```
4.1007.4.34 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::insert(initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the unordered\_multimap.

#### Parameters

|                  |                                                                 |
|------------------|-----------------------------------------------------------------|
| <code>__l</code> | A std::initializer_list<value_type> of elements to be inserted. |
|------------------|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 1093 of file unordered\_map.h.

```
4.1007.4.35 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::key_eq() const [inline]
```

Returns the key comparison object with which the unordered\_multimap was constructed.

Definition at line 1189 of file unordered\_map.h.

```
4.1007.4.36 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::load_factor() const [inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1326 of file unordered\_map.h.

```
4.1007.4.37 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::max_bucket_count() const [inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multimap.

Definition at line 1249 of file unordered\_map.h.

```
4.1007.4.38 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::max_load_factor() const [inline], [noexcept]
```

Returns a positive number that the unordered\_multimap tries to keep the load factor less than or equal to.

Definition at line 1332 of file unordered\_map.h.

```
4.1007.4.39 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::max_load_factor(float __z) [inline]
```

Change the unordered\_multimap maximum load factor.

#### Parameters

|     |                              |
|-----|------------------------------|
| __z | The new maximum load factor. |
|-----|------------------------------|

Definition at line 1340 of file unordered\_map.h.

```
4.1007.4.40 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::max_size() const [inline], [noexcept]
```

Returns the maximum size of the unordered\_multimap.

Definition at line 919 of file unordered\_map.h.

4.1007.4.41 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::operator=( const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]`

Copy assignment operator.

4.1007.4.42 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::operator=( unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]`

Move assignment operator.

4.1007.4.43 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::operator=( initializer_list<value_type> __l ) [inline]`

Unordered\_multimap list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_multimap and that the resulting unordered\_multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 893 of file unordered\_map.h.

4.1007.4.44 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::rehash( size_type __n ) [inline]`

May rehash the unordered\_multimap.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_multimap maximum load factor.

Definition at line 1351 of file unordered\_map.h.

4.1007.4.45 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::reserve( size_type __n ) [inline]`

Prepare the unordered\_multimap for a specified number of elements.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1362 of file unordered\_map.h.

4.1007.4.46 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]`

Returns the size of the unordered\_multimap.

Definition at line 914 of file unordered\_map.h.

4.1007.4.47 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap ( unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & _x ) [inline]`

Swaps data with another unordered\_multimap.

#### Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__x</code> | An unordered_multimap of the same element and allocator types. |
|------------------|----------------------------------------------------------------|

This exchanges the elements between two unordered\_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 1175 of file unordered\_map.h.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 4.1008 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inherits std::\_\_allow\_copy\_cons< bool >.

#### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef allocator\_type::pointer [pointer](#)
- typedef allocator\_type::const\_pointer [const\\_pointer](#)
- typedef allocator\_type::reference [reference](#)
- typedef allocator\_type::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_multiset](#) ([size\\_type](#) \_\_n=10, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [template](#)<[typename](#) \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_f, \_InputIterator \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) (const [unordered\\_multiset](#) &)=default
- [unordered\\_multiset](#) ([unordered\\_multiset](#) &&)=default
- [unordered\\_multiset](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [size\\_type](#) [bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) [bucket\\_count](#) () const noexcept
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_n) const
- [const\\_iterator](#) [cbegin](#) () const noexcept
- [const\\_iterator](#) [cend](#) () const noexcept
- void [clear](#) () noexcept
- [size\\_type](#) [count](#) (const [key\\_type](#) &\_\_x) const
- [template](#)<[typename](#)... \_Args>  
[iterator](#) [emplace](#) (\_Args &&...\_\_args)
- [template](#)<[typename](#)... \_Args>  
[iterator](#) [emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&...\_\_args)
- bool [empty](#) () const noexcept
- [size\\_type](#) [erase](#) (const [key\\_type](#) &\_\_x)
- [iterator](#) [erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [allocator\\_type](#) [get\\_allocator](#) () const noexcept
- [hasher](#) [hash\\_function](#) () const
- [template](#)<[typename](#) \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [key\\_equal](#) [key\\_eq](#) () const
- float [load\\_factor](#) () const noexcept
- [size\\_type](#) [max\\_bucket\\_count](#) () const noexcept
- float [max\\_load\\_factor](#) () const noexcept
- void [max\\_load\\_factor](#) (float \_\_z)
- [size\\_type](#) [max\\_size](#) () const noexcept
- [unordered\\_multiset](#) & [operator=](#) (const [unordered\\_multiset](#) &)=default
- [unordered\\_multiset](#) & [operator=](#) ([unordered\\_multiset](#) &&\_\_x)=default
- [unordered\\_multiset](#) & [operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- void [rehash](#) ([size\\_type](#) \_\_n)
- void [reserve](#) ([size\\_type](#) \_\_n)
- [size\\_type](#) [size](#) () const noexcept
- void [swap](#) ([unordered\\_multiset](#) &\_\_x)
  
- [iterator](#) [begin](#) () noexcept
- [const\\_iterator](#) [begin](#) () const noexcept
  
- [iterator](#) [end](#) () noexcept
- [const\\_iterator](#) [end](#) () const noexcept
  
- [iterator](#) [insert](#) (const [value\\_type](#) &\_\_x)

- [iterator insert](#) ([value\\_type](#) &&\_\_x)
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, const [value\\_type](#) &\_\_x)
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, [value\\_type](#) &&\_\_x)
- [iterator erase](#) ([const\\_iterator](#) \_\_position)
- [iterator erase](#) ([iterator](#) \_\_it)
- [iterator find](#) (const [key\\_type](#) &\_\_x)
- [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
- [std::pair< iterator, iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x)
- [std::pair< const\\_iterator, const\\_iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x) const
- [local\\_iterator begin](#) ([size\\_type](#) \_\_n)
- [const\\_local\\_iterator begin](#) ([size\\_type](#) \_\_n) const
- [const\\_local\\_iterator cbegin](#) ([size\\_type](#) \_\_n) const
- [local\\_iterator end](#) ([size\\_type](#) \_\_n)
- [const\\_local\\_iterator end](#) ([size\\_type](#) \_\_n) const
- [const\\_local\\_iterator cend](#) ([size\\_type](#) \_\_n) const

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 >  
bool operator== (const unordered\_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered\_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &)`

#### 4.1008.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

#### Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__umset_hashtable`.

Definition at line 698 of file `unordered_set.h`.

## 4.1008.2 Member Typedef Documentation

4.1008.2.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::allocator_type`

Public typedefs.

Definition at line 711 of file unordered\_set.h.

4.1008.2.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 721 of file unordered\_set.h.

4.1008.2.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_local_iterator`

Iterator-related typedefs.

Definition at line 723 of file unordered\_set.h.

4.1008.2.4 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_pointer`

Iterator-related typedefs.

Definition at line 717 of file unordered\_set.h.

4.1008.2.5 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_reference`

Iterator-related typedefs.

Definition at line 719 of file unordered\_set.h.

4.1008.2.6 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::difference_type`

Iterator-related typedefs.

Definition at line 725 of file unordered\_set.h.

4.1008.2.7 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher`

Public typedefs.

Definition at line 709 of file unordered\_set.h.

```
4.1008.2.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::iterator
```

Iterator-related typedefs.

Definition at line 720 of file unordered\_set.h.

```
4.1008.2.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::key_equal
```

Public typedefs.

Definition at line 710 of file unordered\_set.h.

```
4.1008.2.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::key_type
```

Public typedefs.

Definition at line 707 of file unordered\_set.h.

```
4.1008.2.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::local_iterator
```

Iterator-related typedefs.

Definition at line 722 of file unordered\_set.h.

```
4.1008.2.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef allocator_type::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::pointer
```

Iterator-related typedefs.

Definition at line 716 of file unordered\_set.h.

```
4.1008.2.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef allocator_type::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::reference
```

Iterator-related typedefs.

Definition at line 718 of file unordered\_set.h.

```
4.1008.2.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::size_type
```

Iterator-related typedefs.

Definition at line 724 of file unordered\_set.h.

4.1008.2.15 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 708 of file unordered\_set.h.

#### 4.1008.3 Constructor & Destructor Documentation

4.1008.3.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset ( size_type __n = 10, const hasher & __hf = hasher (), const key_equal & __eqf = key_equal (), const allocator_type & __a = allocator_type () ) [inline], [explicit]`

Default constructor creates no elements.

##### Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__n</code>   | Initial number of buckets. |
| <code>__hf</code>  | A hash functor.            |
| <code>__eqf</code> | A key equality functor.    |
| <code>__a</code>   | An allocator object.       |

Definition at line 737 of file unordered\_set.h.

4.1008.3.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename _InputIterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset ( _InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher (), const key_equal & __eqf = key_equal (), const allocator_type & __a = allocator_type () ) [inline]`

Builds an unordered\_multiset from a range.

##### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_multiset consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is distance(`__first`,`__last`)).

Definition at line 758 of file unordered\_set.h.

4.1008.3.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset ( const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]`

Copy constructor.

4.1008.3.4 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::unordered_multiset ( unordered_multiset<_Value, _Hash, _Pred, _Alloc> && ) [default]`

Move constructor.

4.1008.3.5 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::unordered_multiset ( initializer_list<value_type> __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() ) [inline]`

Builds an unordered\_multiset from an initializer\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_multiset consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 783 of file unordered\_set.h.

#### 4.1008.4 Member Function Documentation

4.1008.4.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::begin ( ) [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 848 of file unordered\_set.h.

4.1008.4.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 852 of file unordered\_set.h.

4.1008.4.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::begin ( size_type __n ) [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1175 of file unordered\_set.h.

```
4.1008.4.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin
(size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1179 of file unordered\_set.h.

```
4.1008.4.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count ()
const [inline],[noexcept]
```

Returns the number of buckets of the unordered\_multiset.

Definition at line 1141 of file unordered\_set.h.

```
4.1008.4.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin ()
const [inline],[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 875 of file unordered\_set.h.

```
4.1008.4.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::cbegin (size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1183 of file unordered\_set.h.

```
4.1008.4.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend ()
 const [inline],[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

Definition at line 883 of file unordered\_set.h.

```
4.1008.4.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1203 of file unordered\_set.h.

```
4.1008.4.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ()
 [inline],[noexcept]
```

Erases all elements in an unordered\_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1059 of file unordered\_set.h.

```
4.1008.4.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (const
 key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

**Returns**

Number of elements with specified key.

Definition at line 1118 of file unordered\_set.h.

4.1008.4.12 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace ( _Args &&... __args ) [inline]`

Builds and insert an element into the unordered\_multiset.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

#### Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 897 of file unordered\_set.h.

4.1008.4.13 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]`

Inserts an element into the unordered\_multiset.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

#### Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires amortized constant time.

Definition at line 919 of file unordered\_set.h.

4.1008.4.14 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]`

Returns true if the unordered\_multiset is empty.

Definition at line 827 of file unordered\_set.h.

4.1008.4.15 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

Definition at line 862 of file unordered\_set.h.

```
4.1008.4.16 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (_)
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

Definition at line 866 of file unordered\_set.h.

```
4.1008.4.17 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read-only local iterator.

Definition at line 1195 of file unordered\_set.h.

```
4.1008.4.18 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read-only local iterator.

Definition at line 1199 of file unordered\_set.h.

```
4.1008.4.19 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1129 of file unordered\_set.h.

```
4.1008.4.20 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_multiset< _Value,
 _Hash, _Pred, _Alloc >::equal_range (const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1133 of file unordered\_set.h.

```
4.1008.4.21 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an unordered\_multiset.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1005 of file unordered\_set.h.

```
4.1008.4.22 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (iterator __it
) [inline]
```

Erases an element from an unordered\_multiset.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1010 of file unordered\_set.h.

**4.1008.4.23** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::erase ( const key_type & __x ) [inline]`

Erases elements according to the provided key.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1028 of file unordered\_set.h.

**4.1008.4.24** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::erase ( const_iterator __first, const_iterator __last ) [inline]`

Erases a [`__first`,`__last`) range of elements from an unordered\_multiset.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

#### Returns

The iterator `__last`.

This function erases a sequence of elements from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1048 of file unordered\_set.h.

**4.1008.4.25** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::find ( const key_type & __x ) [inline]`

Tries to locate an element in an unordered\_multiset.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 1104 of file unordered\_set.h.

```
4.1008.4.26 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (const
 key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_multiset.

**Parameters**

|     |                        |
|-----|------------------------|
| __x | Element to be located. |
|-----|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 1108 of file unordered\_set.h.

```
4.1008.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
 >::get_allocator () const [inline], [noexcept]
```

Returns the allocator object with which the unordered\_multiset was constructed.

Definition at line 820 of file unordered\_set.h.

```
4.1008.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function ()
 const [inline]
```

Returns the hash functor object with which the unordered\_multiset was constructed.

Definition at line 1080 of file unordered\_set.h.

```
4.1008.4.29 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (const
 value_type & __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|     |                         |
|-----|-------------------------|
| __x | Element to be inserted. |
|-----|-------------------------|

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 931 of file unordered\_set.h.

```
4.1008.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 935 of file unordered\_set.h.

```
4.1008.4.31 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint, const value_type & __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires amortized constant.

Definition at line 957 of file unordered\_set.h.

```
4.1008.4.32 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint, value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

## Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires amortized constant.

Definition at line 961 of file `unordered_set.h`.

```
4.1008.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> template<typename _InputIterator > void std::unordered_multiset< _Value, _Hash,
 _Pred, _Alloc >::insert (_InputIterator __first, _InputIterator __last) [inline]
```

A template function that inserts a range of elements.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 975 of file `unordered_set.h`.

```
4.1008.4.34 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 initializer_list<value_type> & __l) [inline]
```

Inserts a list of elements into the `unordered_multiset`.

## Parameters

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
|------------------|------------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 986 of file `unordered_set.h`.

```
4.1008.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq ()
 const [inline]
```

Returns the key comparison object with which the `unordered_multiset` was constructed.

Definition at line 1086 of file `unordered_set.h`.

4.1008.4.36 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]`

Returns the average number of elements per bucket.

Definition at line 1211 of file unordered\_set.h.

4.1008.4.37 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline], [noexcept]`

Returns the maximum number of buckets of the unordered\_multiset.

Definition at line 1146 of file unordered\_set.h.

4.1008.4.38 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline], [noexcept]`

Returns a positive number that the unordered\_multiset tries to keep the load factor less than or equal to.

Definition at line 1217 of file unordered\_set.h.

4.1008.4.39 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( float __z ) [inline]`

Change the unordered\_multiset maximum load factor.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

Definition at line 1225 of file unordered\_set.h.

4.1008.4.40 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size ( ) const [inline], [noexcept]`

Returns the maximum size of the unordered\_multiset.

Definition at line 837 of file unordered\_set.h.

4.1008.4.41 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= ( const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]`

Copy assignment operator.

4.1008.4.42 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= ( unordered_multiset< _Value, _Hash, _Pred, _Alloc > && __x ) [default]`

Move assignment operator.

4.1008.4.43 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= ( initializer_list< value_type > __l ) [inline]`

Unordered\_multiset list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_multiset with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_multiset and that the resulting unordered\_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 811 of file unordered\_set.h.

4.1008.4.44 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash ( size_type __n ) [inline]`

May rehash the unordered\_multiset.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_multiset maximum load factor.

Definition at line 1236 of file unordered\_set.h.

4.1008.4.45 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve ( size_type __n ) [inline]`

Prepare the unordered\_multiset for a specified number of elements.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1247 of file unordered\_set.h.

4.1008.4.46 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size ( ) const [inline],[noexcept]`

Returns the size of the unordered\_multiset.

Definition at line 832 of file unordered\_set.h.

4.1008.4.47 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap ( unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x ) [inline]`

Swaps data with another unordered\_multiset.

## Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__x</code> | An unordered_multiset of the same element and allocator types. |
|------------------|----------------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 1072 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 4.1009 std::unordered\_set&lt; \_Value, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inherits `std::__allow_copy_cons< bool >`.

## Public Types

- `typedef _Hashtable::key_type` [key\\_type](#)
- `typedef _Hashtable::value_type` [value\\_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key\\_equal](#)
- `typedef _Hashtable::allocator_type` [allocator\\_type](#)
- `typedef allocator_type::pointer` [pointer](#)
- `typedef allocator_type::const_pointer` [const\\_pointer](#)
- `typedef allocator_type::reference` [reference](#)
- `typedef allocator_type::const_reference` [const\\_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const\\_iterator](#)
- `typedef _Hashtable::local_iterator` [local\\_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const\\_local\\_iterator](#)
- `typedef _Hashtable::size_type` [size\\_type](#)
- `typedef _Hashtable::difference_type` [difference\\_type](#)

## Public Member Functions

- [unordered\\_set](#) ([size\\_type](#) \_\_n=10, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- `template<typename _InputIterator >`  
[unordered\\_set](#) (`_InputIterator` \_\_f, `_InputIterator` \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_set](#) (const [unordered\\_set](#) &)=default
- [unordered\\_set](#) ([unordered\\_set](#) &&)=default
- [unordered\\_set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [size\\_type](#) [bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) [bucket\\_count](#) () const noexcept

- [size\\_type bucket\\_size](#) ([size\\_type](#) \_\_n) const
  - [const\\_iterator cbegin](#) () const noexcept
  - [const\\_iterator cend](#) () const noexcept
  - void [clear](#) () noexcept
  - [size\\_type count](#) (const [key\\_type](#) &\_\_x) const
  - template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&...\_\_args)
  - template<typename... \_Args>  
[iterator](#) [emplace\\_hint](#) ([const\\_iterator](#) \_\_pos, \_Args &&...\_\_args)
  - bool [empty](#) () const noexcept
  - [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
  - [iterator erase](#) ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
  - [allocator\\_type get\\_allocator](#) () const noexcept
  - [hasher hash\\_function](#) () const
  - template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void [insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [key\\_equal key\\_eq](#) () const
  - float [load\\_factor](#) () const noexcept
  - [size\\_type max\\_bucket\\_count](#) () const noexcept
  - float [max\\_load\\_factor](#) () const noexcept
  - void [max\\_load\\_factor](#) (float \_\_z)
  - [size\\_type max\\_size](#) () const noexcept
  - [unordered\\_set](#) & [operator=](#) (const [unordered\\_set](#) &)=default
  - [unordered\\_set](#) & [operator=](#) ([unordered\\_set](#) &&)=default
  - [unordered\\_set](#) & [operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - void [rehash](#) ([size\\_type](#) \_\_n)
  - void [reserve](#) ([size\\_type](#) \_\_n)
  - [size\\_type size](#) () const noexcept
  - void [swap](#) ([unordered\\_set](#) &\_\_x)
- 
- [iterator begin](#) () noexcept
  - [const\\_iterator begin](#) () const noexcept
- 
- [iterator end](#) () noexcept
  - [const\\_iterator end](#) () const noexcept
- 
- [std::pair](#)< [iterator](#), bool > [insert](#) (const [value\\_type](#) &\_\_x)
  - [std::pair](#)< [iterator](#), bool > [insert](#) ([value\\_type](#) &&\_\_x)
- 
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, const [value\\_type](#) &\_\_x)
  - [iterator insert](#) ([const\\_iterator](#) \_\_hint, [value\\_type](#) &&\_\_x)
- 
- [iterator erase](#) ([const\\_iterator](#) \_\_position)
  - [iterator erase](#) ([iterator](#) \_\_it)
- 
- [iterator find](#) (const [key\\_type](#) &\_\_x)
  - [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
- 
- [std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const [key\\_type](#) &\_\_x)

- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `local_iterator begin` (size\_type \_\_n)
- `const_local_iterator begin` (size\_type \_\_n) const
- `const_local_iterator cbegin` (size\_type \_\_n) const
- `local_iterator end` (size\_type \_\_n)
- `const_local_iterator end` (size\_type \_\_n) const
- `const_local_iterator cend` (size\_type \_\_n) const

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (const `unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 >` &)

#### 4.1009.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_set< _Value, _Hash, _Pred, _Alloc >`

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

#### Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__uset_hashtable`.

Definition at line 93 of file `unordered_set.h`.

#### 4.1009.2 Member Typedef Documentation

4.1009.2.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::allocator_type`

Public typedefs.

Definition at line 106 of file `unordered_set.h`.

4.1009.2.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 116 of file unordered\_set.h.

```
4.1009.2.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc
 >::const_local_iterator
```

Iterator-related typedefs.

Definition at line 118 of file unordered\_set.h.

```
4.1009.2.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef allocator_type::const_pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc
 >::const_pointer
```

Iterator-related typedefs.

Definition at line 112 of file unordered\_set.h.

```
4.1009.2.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef allocator_type::const_reference std::unordered_set< _Value, _Hash, _Pred, _Alloc
 >::const_reference
```

Iterator-related typedefs.

Definition at line 114 of file unordered\_set.h.

```
4.1009.2.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_set< _Value, _Hash, _Pred, _Alloc
 >::difference_type
```

Iterator-related typedefs.

Definition at line 120 of file unordered\_set.h.

```
4.1009.2.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 104 of file unordered\_set.h.

```
4.1009.2.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 115 of file unordered\_set.h.

```
4.1009.2.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc
 >::key_equal
```

Public typedefs.

Definition at line 105 of file unordered\_set.h.

```
4.1009.2.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::key_type
```

Public typedefs.

Definition at line 102 of file unordered\_set.h.

```
4.1009.2.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::local_iterator
```

Iterator-related typedefs.

Definition at line 117 of file unordered\_set.h.

```
4.1009.2.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef allocator_type::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::pointer
```

Iterator-related typedefs.

Definition at line 111 of file unordered\_set.h.

```
4.1009.2.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef allocator_type::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::reference
```

Iterator-related typedefs.

Definition at line 113 of file unordered\_set.h.

```
4.1009.2.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::size_type
```

Iterator-related typedefs.

Definition at line 119 of file unordered\_set.h.

```
4.1009.2.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::value_type
```

Public typedefs.

Definition at line 103 of file unordered\_set.h.

#### 4.1009.3 Constructor & Destructor Documentation

```
4.1009.3.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (size_type __n
= 10, const hasher & __hf = hasher (), const key_equal & __eqf = key_equal (), const allocator_type & __a =
allocator_type ()) [inline], [explicit]
```

Default constructor creates no elements.

## Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__n</code>   | Initial number of buckets. |
| <code>__hf</code>  | A hash functor.            |
| <code>__eqf</code> | A key equality functor.    |
| <code>__a</code>   | An allocator object.       |

Definition at line 132 of file `unordered_set.h`.

```
4.1009.3.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::unordered_set (_InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher(),
const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_set` from a range.

## Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_set` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 153 of file `unordered_set.h`.

```
4.1009.3.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (const
unordered_set< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

```
4.1009.3.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc
= std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
unordered_set< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

```
4.1009.3.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc
= std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal &
__eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_set` from an `initializer_list`.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An <code>initializer_list</code> . |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_set consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 178 of file unordered\_set.h.

#### 4.1009.4 Member Function Documentation

**4.1009.4.1** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( ) [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 243 of file unordered\_set.h.

**4.1009.4.2** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 247 of file unordered\_set.h.

**4.1009.4.3** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( size_type __n ) [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

##### Returns

A read-only local iterator.

Definition at line 593 of file unordered\_set.h.

**4.1009.4.4** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 597 of file unordered\_set.h.

**4.1009.4.5** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::bucket_count ( ) const [inline], [noexcept]`

Returns the number of buckets of the unordered\_set.

Definition at line 559 of file unordered\_set.h.

**4.1009.4.6** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 270 of file unordered\_set.h.

**4.1009.4.7** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cbegin ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 601 of file unordered\_set.h.

**4.1009.4.8** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cend ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

Definition at line 278 of file unordered\_set.h.

**4.1009.4.9** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cend ( size_type __n ) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 621 of file unordered\_set.h.

**4.1009.4.10** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::clear ( ) [inline], [noexcept]`

Erases all elements in an unordered\_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 471 of file unordered\_set.h.

**4.1009.4.11** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::count ( const key_type & __x ) const [inline]`

Finds the number of elements.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

**Returns**

Number of elements with specified key.

This function only makes sense for unordered\_multisets; for unordered\_set the result will either be 0 (not present) or 1 (present).

Definition at line 534 of file unordered\_set.h.

**4.1009.4.12** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc>::emplace ( _Args &&... __args ) [inline]`

Attempts to build and insert an element into the unordered\_set.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 300 of file unordered\_set.h.

4.1009.4.13 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]`

Attempts to insert an element into the unordered\_set.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

#### Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires amortized constant time.

Definition at line 326 of file `unordered_set.h`.

4.1009.4.14 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> bool std::unordered_set<_Value, _Hash, _Pred, _Alloc>::empty ( ) const [inline], [noexcept]`

Returns true if the unordered\_set is empty.

Definition at line 222 of file `unordered_set.h`.

4.1009.4.15 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( ) [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

Definition at line 257 of file `unordered_set.h`.

4.1009.4.16 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( ) const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

Definition at line 261 of file `unordered_set.h`.

4.1009.4.17 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( size_type __n ) [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read-only local iterator.

Definition at line 613 of file unordered\_set.h.

```
4.1009.4.18 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read-only local iterator.

Definition at line 617 of file unordered\_set.h.

```
4.1009.4.19 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_set< _Value, _Hash, _Pred, _Alloc
 >::equal_range (const key_type & __x) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 547 of file unordered\_set.h.

```
4.1009.4.20 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_set< _Value, _Hash,
 _Pred, _Alloc >::equal_range (const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 551 of file unordered\_set.h.

```
4.1009.4.21 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (const_iterator
 __position) [inline]
```

Erases an element from an unordered\_set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 421 of file unordered\_set.h.

```
4.1009.4.22 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (iterator __it)
 [inline]
```

Erases an element from an unordered\_set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 426 of file unordered\_set.h.

```
4.1009.4.23 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (const
 key_type & __x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_set`. For an `unordered_set` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 443 of file `unordered_set.h`.

```
4.1009.4.24 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (const_iterator
 __first, const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_set`.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 461 of file `unordered_set.h`.

```
4.1009.4.25 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (const key_type &
 __x) [inline]
```

Tries to locate an element in an `unordered_set`.

## Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 516 of file unordered\_set.h.

```
4.1009.4.26 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (const
 key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_set.

**Parameters**

|     |                        |
|-----|------------------------|
| __x | Element to be located. |
|-----|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 520 of file unordered\_set.h.

```
4.1009.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator ()
 const [inline], [noexcept]
```

Returns the allocator object with which the unordered\_set was constructed.

Definition at line 215 of file unordered\_set.h.

```
4.1009.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function () const
 [inline]
```

Returns the hash functor object with which the unordered\_set was constructed.

Definition at line 492 of file unordered\_set.h.

```
4.1009.4.29 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert
 (const value_type & __x) [inline]
```

Attempts to insert an element into the unordered\_set.

**Parameters**

|     |                         |
|-----|-------------------------|
| __x | Element to be inserted. |
|-----|-------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 344 of file unordered\_set.h.

```
4.1009.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert
 (value_type && __x) [inline]
```

Attempts to insert an element into the unordered\_set.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 348 of file unordered\_set.h.

```
4.1009.4.31 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (const_iterator
 __hint, const value_type & __x) [inline]
```

Attempts to insert an element into the unordered\_set.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires amortized constant.

Definition at line 373 of file `unordered_set.h`.

```
4.1009.4.32 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (const_iterator
 __hint, value_type && __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.-html>

Insertion requires amortized constant.

Definition at line 377 of file `unordered_set.h`.

```
4.1009.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
 std::allocator<_Value>> template<typename _InputIterator > void std::unordered_set< _Value, _Hash, _Pred,
 _Alloc >::insert (_InputIterator __first, _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 392 of file `unordered_set.h`.

4.1009.4.34 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert ( initializer_list<value_type> & __l ) [inline]`

Attempts to insert a list of elements into the unordered\_set.

#### Parameters

|                  |                                                                 |
|------------------|-----------------------------------------------------------------|
| <code>__l</code> | A std::initializer_list<value_type> of elements to be inserted. |
|------------------|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 403 of file unordered\_set.h.

4.1009.4.35 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> key_equal std::unordered_set<_Value, _Hash, _Pred, _Alloc>::key_eq ( ) const [inline]`

Returns the key comparison object with which the unordered\_set was constructed.

Definition at line 498 of file unordered\_set.h.

4.1009.4.36 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float std::unordered_set<_Value, _Hash, _Pred, _Alloc>::load_factor ( ) const [inline], [noexcept]`

Returns the average number of elements per bucket.

Definition at line 629 of file unordered\_set.h.

4.1009.4.37 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_bucket_count ( ) const [inline], [noexcept]`

Returns the maximum number of buckets of the unordered\_set.

Definition at line 564 of file unordered\_set.h.

4.1009.4.38 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( ) const [inline], [noexcept]`

Returns a positive number that the unordered\_set tries to keep the load factor less than or equal to.

Definition at line 635 of file unordered\_set.h.

4.1009.4.39 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( float __z ) [inline]`

Change the unordered\_set maximum load factor.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

Definition at line 643 of file unordered\_set.h.

4.1009.4.40 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_size ( ) const [inline], [noexcept]`

Returns the maximum size of the unordered\_set.

Definition at line 232 of file unordered\_set.h.

4.1009.4.41 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator= ( const unordered_set<_Value, _Hash, _Pred, _Alloc> & ) [default]`

Copy assignment operator.

4.1009.4.42 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator= ( unordered_set<_Value, _Hash, _Pred, _Alloc> && ) [default]`

Move assignment operator.

4.1009.4.43 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator= ( initializer_list<value_type> __l ) [inline]`

Unordered\_set list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_set and that the resulting unordered\_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 206 of file unordered\_set.h.

4.1009.4.44 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::rehash ( size_type __n ) [inline]`

May rehash the unordered\_set.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_set maximum load factor.

Definition at line 654 of file unordered\_set.h.

4.1009.4.45 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::reserve ( size_type __n ) [inline]`

Prepare the unordered\_set for a specified number of elements.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 665 of file `unordered_set.h`.

**4.1009.4.46** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]`

Returns the size of the `unordered_set`.

Definition at line 227 of file `unordered_set.h`.

**4.1009.4.47** `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap ( unordered_set< _Value, _Hash, _Pred, _Alloc > & __x ) [inline]`

Swaps data with another `unordered_set`.

## Parameters

|                  |                                                                        |
|------------------|------------------------------------------------------------------------|
| <code>__x</code> | An <code>unordered_set</code> of the same element and allocator types. |
|------------------|------------------------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

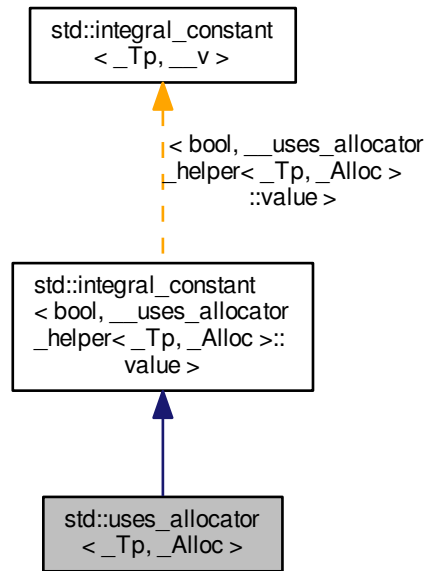
Definition at line 484 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 4.1010 std::uses\_allocator&lt;\_Tp, \_Alloc&gt; Struct Template Reference

Inheritance diagram for std::uses\_allocator<\_Tp, \_Alloc>:



## Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `bool` **value**

## 4.1010.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::uses_allocator<_Tp, _Alloc>
```

[allocator.uses.trait]

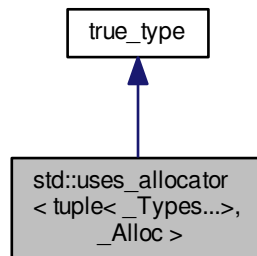
Definition at line 58 of file `uses_allocator.h`.

The documentation for this struct was generated from the following file:

- `uses_allocator.h`

#### 4.1011 `std::uses_allocator< tuple< _Types...>, _Alloc >` Struct Template Reference

Inheritance diagram for `std::uses_allocator< tuple< _Types...>, _Alloc >`:



##### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** ()

##### Static Public Attributes

- static constexpr `_Tp` **value**

##### 4.1011.1 Detailed Description

`template<typename... _Types, typename _Alloc> struct std::uses_allocator< tuple< _Types...>, _Alloc >`

Partial specialization for tuples.

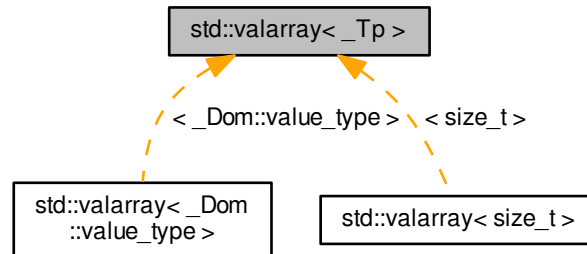
Definition at line 1066 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.1012 std::valarray&lt;\_Tp&gt; Class Template Reference

Inheritance diagram for std::valarray<\_Tp>:



## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- `valarray` ()
- `valarray` (size\_t)
- `valarray` (const \_Tp &, size\_t)
- `valarray` (const \_Tp \*\_\_restrict\_\_, size\_t)
- `valarray` (const `valarray` &)
- `valarray` (`valarray` &&) noexcept
- `valarray` (const `slice_array`<\_Tp> &)
- `valarray` (const `gslice_array`<\_Tp> &)
- `valarray` (const `mask_array`<\_Tp> &)
- `valarray` (const `indirect_array`<\_Tp> &)
- `valarray` (`initializer_list`<\_Tp>)
- template<class \_Dom>
  - `valarray` (const \_Expr<\_Dom, \_Tp> &\_\_e)
- template<typename \_Tp>
  - `valarray` (const \_Tp \*\_\_restrict\_\_ \_\_p, size\_t \_\_n)
- \_Expr<\_ValFunClos<\_ValArray, \_Tp>, \_Tp> `apply` (\_Tp func(\_Tp)) const
- \_Expr<\_RefFunClos<\_ValArray, \_Tp>, \_Tp> `apply` (\_Tp func(const \_Tp &)) const
- `valarray`<\_Tp> `cshift` (int \_\_n) const
- \_Tp `max` () const
- \_Tp `min` () const
- \_UnaryOp<\_\_logical\_not>::Rt `operator!` () const
- `valarray`<\_Tp> & `operator%=>` (const \_Tp &)
- `valarray`<\_Tp> & `operator%=>` (const `valarray`<\_Tp> &)

- template<class \_Dom >  
valarray<\_Tp> & operator%= (const \_Expr<\_Dom, \_Tp> &)
- valarray<\_Tp> & operator&= (const \_Tp &)
- valarray<\_Tp> & operator&= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator&= (const \_Expr<\_Dom, \_Tp> &)
- valarray<\_Tp> & operator\*= (const \_Tp &)
- valarray<\_Tp> & operator\*= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator\*= (const \_Expr<\_Dom, \_Tp> &)
- \_UnaryOp<\_\_unary\_plus>::Rt operator+ () const
- valarray<\_Tp> & operator+= (const \_Tp &)
- valarray<\_Tp> & operator+= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator+= (const \_Expr<\_Dom, \_Tp> &)
- \_UnaryOp<\_\_negate>::Rt operator- () const
- valarray<\_Tp> & operator-= (const \_Tp &)
- valarray<\_Tp> & operator-= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator-= (const \_Expr<\_Dom, \_Tp> &)
- valarray<\_Tp> & operator/= (const \_Tp &)
- valarray<\_Tp> & operator/= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator/= (const \_Expr<\_Dom, \_Tp> &)
- valarray<\_Tp> & operator<=<= (const \_Tp &)
- valarray<\_Tp> & operator<=<= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator<=<= (const \_Expr<\_Dom, \_Tp> &)
- valarray<\_Tp> & operator= (const valarray<\_Tp> &\_\_v)
- valarray<\_Tp> & operator= (valarray<\_Tp> &&\_\_v) noexcept
- valarray<\_Tp> & operator= (const \_Tp &\_\_t)
- valarray<\_Tp> & operator= (const slice\_array<\_Tp> &\_\_sa)
- valarray<\_Tp> & operator= (const gslice\_array<\_Tp> &\_\_ga)
- valarray<\_Tp> & operator= (const mask\_array<\_Tp> &\_\_ma)
- valarray<\_Tp> & operator= (const indirect\_array<\_Tp> &\_\_ia)
- valarray & operator= (initializer\_list<\_Tp> \_\_l)
- template<class \_Dom >  
valarray<\_Tp> & operator= (const \_Expr<\_Dom, \_Tp> &)
- valarray<\_Tp> & operator>=>= (const \_Tp &)
- valarray<\_Tp> & operator>=>= (const valarray<\_Tp> &)
- template<class \_Dom >  
valarray<\_Tp> & operator>=>= (const \_Expr<\_Dom, \_Tp> &)
- \_Tp & operator[] (size\_t \_\_i)
- const \_Tp & operator[] (size\_t) const
- \_Expr<\_SClos<\_ValArray, \_Tp>  
\_, \_Tp> operator[] (slice \_\_s) const
- slice\_array<\_Tp> operator[] (slice \_\_s)
- \_Expr<\_GClos<\_ValArray, \_Tp>  
\_, \_Tp> operator[] (const gslice &\_\_s) const
- gslice\_array<\_Tp> operator[] (const gslice &\_\_s)
- valarray<\_Tp> operator[] (const valarray<bool> &\_\_m) const

- [mask\\_array](#)<\_Tp> [operator\[\]](#) (const [valarray](#)<bool> &\_\_m)
- [\\_Expr](#)<\_IClos<\_ValArray, \_Tp>, \_Tp> [operator\[\]](#) (const [valarray](#)<size\_t> &\_\_i) const
- [indirect\\_array](#)<\_Tp> [operator\[\]](#) (const [valarray](#)<size\_t> &\_\_i)
- [valarray](#)<\_Tp> & [operator^](#)= (const \_Tp &)
- [valarray](#)<\_Tp> & [operator^](#)= (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
[valarray](#)<\_Tp> & [operator^](#)= (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & [operator|](#)= (const \_Tp &)
- [valarray](#)<\_Tp> & [operator|](#)= (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
[valarray](#)<\_Tp> & [operator|](#)= (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [\\_UnaryOp](#)<\_\_bitwise\_not>::Rt [operator~](#) () const
- void [resize](#) (size\_t \_\_size, \_Tp \_\_c=\_Tp())
- [valarray](#)<\_Tp> [shift](#) (int \_\_n) const
- size\_t [size](#) () const
- \_Tp [sum](#) () const
- void [swap](#) ([valarray](#)<\_Tp> &\_\_v) noexcept

#### Friends

- class [\\_Array](#)<\_Tp>

#### 4.1012.1 Detailed Description

template<class \_Tp>class std::valarray<\_Tp>

Smart array designed to support numeric processing.

A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

#### Template Parameters

|                     |                              |
|---------------------|------------------------------|
| <a href="#">_Tp</a> | Type of object in the array. |
|---------------------|------------------------------|

Definition at line 116 of file valarray.

#### 4.1012.2 Constructor & Destructor Documentation

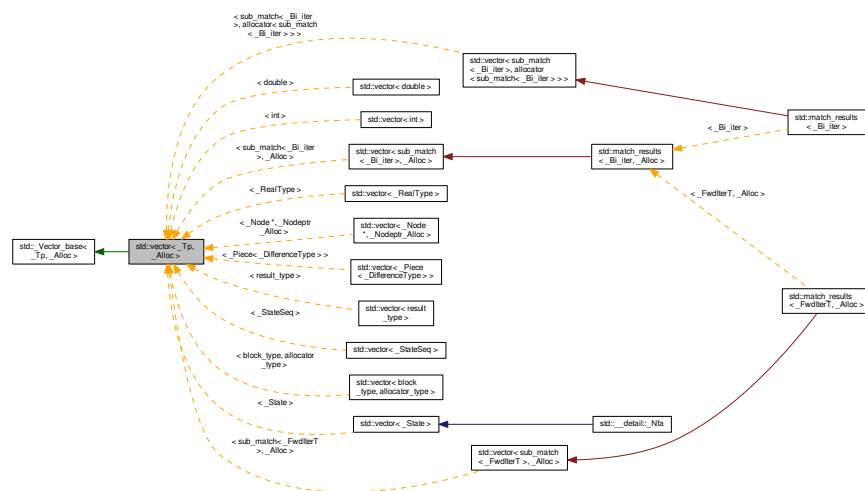
4.1012.2.1 template<class \_Tp> std::valarray<\_Tp>::valarray ( const \_Tp \* \_\_restrict\_\_, size\_t )

Construct an array initialized to the first *n* elements of *t*.

The documentation for this class was generated from the following file:

- [valarray](#)

Inheritance diagram for `std::vector<_Tp, _Alloc>`:



- typedef `_Alloc allocator_type`
- typedef  
`__gnu_cxx::__normal_iterator`  
`< const_pointer, vector > const_iterator`
- typedef  
`_Alloc_traits::const_pointer const_pointer`
- typedef  
`_Alloc_traits::const_reference const_reference`
- typedef `std::reverse_iterator`  
`< const_iterator > const_reverse_iterator`
- typedef `ptrdiff_t difference_type`
- typedef  
`__gnu_cxx::__normal_iterator`  
`< pointer, vector > iterator`
- typedef `_Base::pointer pointer`
- typedef `_Alloc_traits::reference reference`
- typedef `std::reverse_iterator`  
`< iterator > reverse_iterator`
- typedef `size_t size_type`
- typedef `_Tp value_type`

- `vector ()`
- `vector (const allocator_type &__a)`
- `vector (size_type __n, const allocator_type &__a=allocator_type())`
- `vector (size_type __n, const value_type & __value, const allocator_type & __a=allocator_type())`

- [vector](#) (const [vector](#) &\_\_x)
- [vector](#) ([vector](#) &&\_\_x) noexcept
- [vector](#) (const [vector](#) &\_\_x, const allocator\_type &\_\_a)
- [vector](#) ([vector](#) &&\_\_rv, const allocator\_type &\_\_m)
- [vector](#) (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
[vector](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- [~vector](#) () noexcept
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (initializer\_list< value\_type > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) ()
- const\_reference [back](#) () const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- \_Tp \* [data](#) () noexcept
- const \_Tp \* [data](#) () const noexcept
- template<typename... \_Args>  
iterator [emplace](#) (iterator \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const\_iterator [end](#) () const noexcept
- iterator [erase](#) (iterator \_\_position)
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- iterator [insert](#) (iterator \_\_position, const value\_type &\_\_x)
- iterator [insert](#) (iterator \_\_position, value\_type &&\_\_x)
- void [insert](#) (iterator \_\_position, initializer\_list< value\_type > \_\_l)
- void [insert](#) (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void [insert](#) (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type [max\\_size](#) () const noexcept
- [vector](#) & [operator=](#) (const [vector](#) &\_\_x)
- [vector](#) & [operator=](#) ([vector](#) &&\_\_x) noexcept(\_Alloc\_traits
- [vector](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)
- reference [operator\[\]](#) (size\_type \_\_n)
- const\_reference [operator\[\]](#) (size\_type \_\_n) const
- void [pop\\_back](#) ()
- void [push\\_back](#) (const value\_type &\_\_x)

- void **push\_back** (value\_type &&\_\_x)
- [reverse\\_iterator rbegin](#) () noexcept
- [const\\_reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- void [reserve](#) (size\_type \_\_n)
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [shrink\\_to\\_fit](#) ()
- size\_type [size](#) () const noexcept
- void [swap](#) (vector &\_\_x) noexcept([\\_Alloc\\_traits](#))

#### Protected Member Functions

- pointer **\_M\_allocate** (size\_t \_\_n)
- template<typename \_ForwardIterator >  
pointer [\\_M\\_allocate\\_and\\_copy](#) (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_InputIterator >  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- void **\_M\_default\_append** (size\_type \_\_n)
- void **\_M\_default\_initialize** (size\_type \_\_n)
- template<typename... \_Args>  
void **\_M\_emplace\_back\_aux** (\_Args &&...\_\_args)
- void **\_M\_erase\_at\_end** (pointer \_\_pos)
- void **\_M\_fill\_assign** (size\_type \_\_n, const value\_type &\_\_val)
- void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_value)
- void **\_M\_fill\_insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- template<typename \_Integer >  
void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename... \_Args>  
void **\_M\_insert\_aux** (iterator \_\_position, \_Args &&...\_\_args)
- template<typename \_Integer >  
void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void [\\_M\\_range\\_check](#) (size\_type \_\_n) const
- template<typename \_InputIterator >  
void **\_M\_range\_initialize** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))

- template<typename \_ForwardIterator >  
void **\_M\_range\_initialize** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator >  
void **\_M\_range\_insert** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_range\_insert** (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- bool **\_M\_shrink\_to\_fit** ()
- allocator\_type **get\_allocator** () const noexcept

#### Protected Attributes

- \_Vector\_impl **\_M\_impl**

#### 4.1013.1 Detailed Description

template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>>class std::vector< \_Tp, \_Alloc >

A standard container which offers fixed time access to individual elements in any order.

#### Template Parameters

|               |                                             |
|---------------|---------------------------------------------|
| <b>_Tp</b>    | Type of element.                            |
| <b>_Alloc</b> | Allocator type, defaults to allocator<_Tp>. |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of push\_front and pop\_front.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( []) access is also provided as with C-style arrays.

Definition at line 210 of file stl\_vector.h.

#### 4.1013.2 Constructor & Destructor Documentation

4.1013.2.1 template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> std::vector< \_Tp, \_Alloc >::vector ( )  
[inline]

Default constructor creates no elements.

Definition at line 248 of file stl\_vector.h.

4.1013.2.2 template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> std::vector< \_Tp, \_Alloc >::vector ( const allocator\_type & \_\_a ) [inline], [explicit]

Creates a vector with no elements.

#### Parameters

|            |                      |
|------------|----------------------|
| <b>__a</b> | An allocator object. |
|------------|----------------------|

Definition at line 256 of file stl\_vector.h.

**4.1013.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( size_type __n, const allocator_type & __a = allocator_type() ) [inline],[explicit]`

Creates a vector with default constructed elements.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
| <code>__a</code> | An allocator.                               |

This constructor fills the vector with `__n` default constructed elements.

Definition at line 269 of file `stl_vector.h`.

**4.1013.2.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( size_type __n, const value_type & __value, const allocator_type & __a = allocator_type() ) [inline]`

Creates a vector with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the vector with `__n` copies of `__value`.

Definition at line 281 of file `stl_vector.h`.

**4.1013.2.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( const vector<_Tp, _Alloc> & __x ) [inline]`

Vector copy constructor.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

The newly-created vector uses a copy of the allocation object used by `__x`. All the elements of `__x` are copied, but any extra memory in `__x` (for fast expansion) will not be copied.

Definition at line 310 of file `stl_vector.h`.

**4.1013.2.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( vector<_Tp, _Alloc> && __x ) [inline],[noexcept]`

Vector move constructor.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

The newly-created vector contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified vector.

Definition at line 327 of file `stl_vector.h`.

4.1013.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( const vector<_Tp, _Alloc> & __x, const allocator_type & __a ) [inline]`

Copy constructor with alternative allocator.

Definition at line 331 of file stl\_vector.h.

4.1013.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( vector<_Tp, _Alloc> && __rv, const allocator_type & __m ) [inline]`

Move constructor with alternative allocator.

Definition at line 340 of file stl\_vector.h.

4.1013.2.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ( initializer_list<value_type> & __l, const allocator_type & __a = allocator_type() ) [inline]`

Builds a vector from an initializer list.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
| <code>__a</code> | An allocator.        |

Create a vector consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 364 of file stl\_vector.h.

4.1013.2.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> std::vector<_Tp, _Alloc>::vector ( _InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type() ) [inline]`

Builds a vector from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |
| <code>__a</code>     | An allocator.      |

Create a vector consisting of copies of the elements from [first,last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 392 of file stl\_vector.h.

4.1013.2.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::~vector ( ) [inline], [noexcept]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 414 of file stl\_vector.h.

## 4.1013.3 Member Function Documentation

4.1013.3.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>> template<typename _ForwardIterator> pointer  
std::vector<_Tp, _Alloc>::M_allocate_and_copy ( size_type __n, _ForwardIterator __first, _ForwardIterator __last )  
[inline], [protected]`

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

Definition at line 1135 of file stl\_vector.h.

4.1013.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>> void std::vector<_Tp, _Alloc>::M_range_check ( size_type __n ) const [inline], [protected]`

Safety check used only from at().

Definition at line 791 of file stl\_vector.h.

Referenced by std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > >>::at().

4.1013.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>> void std::vector<_Tp, _Alloc>::assign ( size_type __n, const value_type & __val ) [inline]`

Assigns a given value to a vector.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 479 of file stl\_vector.h.

Referenced by std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > >>::operator=().

4.1013.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>> void std::vector<_Tp, _Alloc>::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Assigns a range to a vector.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a vector with copies of the elements in the range [`__first`,`__last`).

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 498 of file stl\_vector.h.

4.1013.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>> void std::vector<_Tp, _Alloc>::assign ( initializer_list< value_type > __l ) [inline]`

Assigns an initializer list to a vector.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 524 of file `std_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::assign()`.

**4.1013.3.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::at (`  
`size_type __n ) [inline]`

Provides access to the data contained in the vector.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

## Returns

Read/write reference to data.

## Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 810 of file `std_vector.h`.

**4.1013.3.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc >::at (`  
`size_type __n ) const [inline]`

Provides access to the data contained in the vector.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

## Returns

Read-only (constant) reference to data.

## Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 828 of file `std_vector.h`.

**4.1013.3.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc>::back ( )`  
`[inline]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 855 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution<_RealType>::max()`, and `std::piecewise_linear_distribution<_RealType>::max()`.

**4.1013.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::back ( ) const` `[inline]`

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 863 of file `stl_vector.h`.

**4.1013.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::begin ( )`  
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 538 of file `stl_vector.h`.

Referenced by `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::crend()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::empty()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::front()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::operator<()`, `std::vector<_Tp, _Alloc>::operator=()`, `std::operator==()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::rend()`, and `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::vector()`.

**4.1013.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::begin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file `stl_vector.h`.

**4.1013.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::capacity ( ) const` `[inline], [noexcept]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 725 of file `stl_vector.h`.

**4.1013.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::cbegin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 611 of file `stl_vector.h`.

**4.1013.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::cend ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file `stl_vector.h`.

4.1013.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::clear ( )`  
`[inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1125 of file `stl_vector.h`.

4.1013.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::crbegin ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 629 of file `stl_vector.h`.

4.1013.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::crend ( ) const` `[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file `stl_vector.h`.

4.1013.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> _Tp* std::vector<_Tp, _Alloc>::data ( )`  
`[inline], [noexcept]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 878 of file `stl_vector.h`.

4.1013.3.19 `template<typename _Tp, typename _Alloc> template<typename... _Args> vector<_Tp, _Alloc>::iterator`  
`vector::emplace ( iterator __position, _Args &&... __args )`

Inserts an object in vector before specified iterator.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__args</code>     | Arguments.                   |

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.`

Definition at line 292 of file `vector.tcc`.

References `std::begin()`, and `std::end()`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > >>::insert()`.

4.1013.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::vector<_Tp, _Alloc>::empty ( )`  
`const [inline], [noexcept]`

Returns true if the vector is empty. (Thus `begin()` would equal `end()`.)

Definition at line 734 of file stl\_vector.h.

Referenced by std::piecewise\_constant\_distribution< \_RealType >::densities(), std::piecewise\_linear\_distribution< \_RealType >::densities(), std::piecewise\_constant\_distribution< \_RealType >::intervals(), std::piecewise\_linear\_distribution< \_RealType >::intervals(), std::discrete\_distribution< \_IntType >::max(), std::piecewise\_constant\_distribution< \_RealType >::max(), std::piecewise\_linear\_distribution< \_RealType >::max(), std::piecewise\_constant\_distribution< \_RealType >::min(), std::piecewise\_linear\_distribution< \_RealType >::min(), and std::discrete\_distribution< \_IntType >::probabilities().

**4.1013.3.21** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> iterator std::vector< \_Tp, \_Alloc >::end ( )  
[inline], [noexcept]

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file stl\_vector.h.

Referenced by std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::back(), std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::cbegin(), std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::empty(), \_\_gnu\_parallel::multiseq\_partition(), \_\_gnu\_parallel::multiseq\_selection(), \_\_gnu\_parallel::multiway\_merge\_exact\_splitting(), std::operator<(), std::vector< \_Tp, \_Alloc >::operator=(), std::operator==( ), std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::push\_back(), std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::rbegin(), std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::resize(), and std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::vector().

**4.1013.3.22** template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> const\_iterator std::vector< \_Tp, \_Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file stl\_vector.h.

**4.1013.3.23** template<typename \_Tp, typename \_Alloc > vector< \_Tp, \_Alloc >::iterator vector::erase ( iterator \_\_position )

Remove element at given position.

#### Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <u>__position</u> | Iterator pointing to element to be erased. |
|-------------------|--------------------------------------------|

#### Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 135 of file vector.tcc.

References std::end().

**4.1013.3.24** template<typename \_Tp, typename \_Alloc > vector< \_Tp, \_Alloc >::iterator vector::erase ( iterator \_\_first, iterator \_\_last )

Remove a range of elements.

## Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

## Returns

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 147 of file `vector.tcc`.

References `std::end()`.

**4.1013.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc>::front ( )`  
`[inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 839 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::data()`, `std::piecewise_constant_distribution< _RealType >::min()`, and `std::piecewise_linear_distribution< _RealType >::min()`.

**4.1013.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::front ( ) const` `[inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 847 of file `stl_vector.h`.

**4.1013.3.27** `template<typename _Tp, typename _Alloc > vector<_Tp, _Alloc>::iterator vector::insert ( iterator __position, const value_type & __x )`

Inserts given value into vector before specified iterator.

## Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__x</code>        | Data to be inserted.         |

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 108 of file `vector.tcc`.

References `std::begin()`, and `std::end()`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize()`.

4.1013.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into vector before specified iterator.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__x</code>        | Data to be inserted.         |

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 988 of file `stl_vector.h`.

4.1013.3.29 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::insert ( iterator __position, initializer_list< value_type > __l ) [inline]`

Inserts an `initializer_list` into the vector.

#### Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the vector.       |
| <code>__l</code>        | An <code>initializer_list</code> . |

This function will insert copies of the data in the `initializer_list l` into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1005 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`.

4.1013.3.30 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::insert ( iterator __position, size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the vector.

#### Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the vector.       |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1023 of file `stl_vector.h`.

4.1013.3.31 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> void std::vector<_Tp, _Alloc>::insert ( iterator __position, _InputIterator __first, _InputIterator __last ) [inline]`

Inserts a range into the vector.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__first</code>    | An input iterator.           |
| <code>__last</code>     | An input iterator.           |

This function will insert copies of the data in the range [`__first`,`__last`) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1044 of file `stl_vector.h`.

4.1013.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::max_size ( ) const [inline], [noexcept]`

Returns the `size()` of the largest possible vector.

Definition at line 650 of file `stl_vector.h`.

4.1013.3.33 `template<typename _Tp, typename _Alloc> vector<_Tp, _Alloc> & vector::operator= ( const vector<_Tp, _Alloc> & __x )`

Vector assignment operator.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any extra memory in `__x` (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

Definition at line 161 of file `vector.tcc`.

References `std::_Destroy()`, `std::begin()`, `std::vector<_Tp, _Alloc>::begin()`, `std::end()`, `std::vector<_Tp, _Alloc>::end()`, `std::vector<_Tp, _Alloc>::size()`, and `std::size()`.

4.1013.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector<_Tp, _Alloc>::operator= ( vector<_Tp, _Alloc> && __x ) [inline], [noexcept]`

Vector move assignment operator.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

The contents of `__x` are moved into this vector (without copying, if the allocators permit it). `__x` is a valid, but unspecified vector.

Definition at line 439 of file `stl_vector.h`.

**4.1013.335** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector<_Tp, _Alloc >::operator=(  
initializer_list<value_type> __l) [inline]`

Vector list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 461 of file `std_vector.h`.

**4.1013.336** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc >::operator[](  
size_type __n) [inline]`

Subscript access to the data contained in the vector.

#### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

#### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 770 of file `std_vector.h`.

**4.1013.337** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc  
>::operator[](size_type __n) const [inline]`

Subscript access to the data contained in the vector.

#### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 785 of file `std_vector.h`.

**4.1013.338** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc >::pop_back()  
[inline]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop\_back() is called.

Definition at line 937 of file stl\_vector.h.

**4.1013.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::push_back (const value_type & __x) [inline]`

Add data to the end of the vector.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 901 of file stl\_vector.h.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**4.1013.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 574 of file stl\_vector.h.

**4.1013.3.41** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file stl\_vector.h.

**4.1013.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file stl\_vector.h.

**4.1013.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file stl\_vector.h.

**4.1013.3.44** `template<typename _Tp, typename _Alloc> void vector::reserve ( size_type __n )`

Attempt to preallocate enough memory for specified number of elements.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

## Exceptions

|                                |                                               |
|--------------------------------|-----------------------------------------------|
| <code>std::length_error</code> | If <i>n</i> exceeds <code>max_size()</code> . |
|--------------------------------|-----------------------------------------------|

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 66 of file `vector.tcc`.

References `std::_Destroy()`, and `std::size()`.

**4.1013.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::resize ( size_type __new_size ) [inline]`

Resizes the vector to the specified number of elements.

## Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain. |
|-------------------------|-----------------------------------------------|

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 664 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::multiway_merge_exact_splitting()`, and `__gnu_parallel::parallel_sort_mwms()`.

**4.1013.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::resize ( size_type __new_size, const value_type & __x ) [inline]`

Resizes the vector to the specified number of elements.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain.     |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 684 of file `stl_vector.h`.

**4.1013.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::shrink_to_fit ( ) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 716 of file `stl_vector.h`.

**4.1013.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector< _Tp, _Alloc >::size ( ) const [inline], [noexcept]`

Returns the number of elements in the vector.

Definition at line 645 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::__M_range_check()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution< _IntType >::max()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize()`.

4.1013.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::swap (vector< _Tp, _Alloc > & __x) [inline], [noexcept]`

Swaps data with another vector.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A vector of the same element and allocator types. |
|------------------|---------------------------------------------------|

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 1108 of file `stl_vector.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 4.1014 std::vector< bool, \_Alloc > Class Template Reference

Inherits `std::_Bvector_base< _Alloc >`.

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef [std::reverse\\_iterator](#) `< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef [std::reverse\\_iterator](#) `< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**

#### Public Member Functions

- **vector** (`const allocator_type &__a`)
- **vector** (`size_type __n, const allocator_type &__a=allocator_type()`)
- **vector** (`size_type __n, const bool &__value, const allocator_type &__a=allocator_type()`)
- **vector** (`const vector &__x`)

- **vector** ([vector](#) &&\_\_x) noexcept
- **vector** ([initializer\\_list](#)< bool > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  **vector** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- void **assign** (size\_type \_\_n, const bool &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** ([initializer\\_list](#)< bool > \_\_l)
- reference **at** (size\_type \_\_n)
- const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- const\_reference **back** () const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- size\_type **capacity** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- void **data** () noexcept
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- void **flip** () noexcept
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_position, const bool &\_\_x=bool())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- void **insert** (iterator \_\_p, [initializer\\_list](#)< bool > \_\_l)
- size\_type **max\_size** () const noexcept
- [vector](#) & **operator=** (const [vector](#) &\_\_x)
- [vector](#) & **operator=** ([vector](#) &&\_\_x)
- [vector](#) & **operator=** ([initializer\\_list](#)< bool > \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (bool \_\_x)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size, bool \_\_x=bool())
- void **shrink\_to\_fit** ()
- size\_type **size** () const noexcept
- void **swap** ([vector](#) &\_\_x)

## Static Public Member Functions

- static void **swap** (reference \_\_x, reference \_\_y) noexcept

## Protected Types

- typedef \_Alloc::template  
rebind< \_Bit\_type >::other **\_Bit\_alloc\_type**

## Protected Member Functions

- \_Bit\_type \* **\_M\_allocate** (size\_t \_\_n)
- template<typename \_InputIterator >  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<class \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- iterator **\_M\_copy\_aligned** (const\_iterator \_\_first, const\_iterator \_\_last, iterator \_\_result)
- void **\_M\_deallocate** ()
- void **\_M\_erase\_at\_end** (iterator \_\_pos)
- void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () noexcept
- const \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () const noexcept
- void **\_M\_initialize** (size\_type \_\_n)
- template<typename \_Integer >  
void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_InputIterator >  
void **\_M\_initialize\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_initialize\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void **\_M\_insert\_aux** (iterator \_\_position, bool \_\_x)
- template<typename \_Integer >  
void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_InputIterator >  
void **\_M\_insert\_range** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_insert\_range** (iterator \_\_position, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void **\_M\_range\_check** (size\_type \_\_n) const
- void **\_M\_reallocate** (size\_type \_\_n)
- bool **\_M\_shrink\_to\_fit** ()

## Static Protected Member Functions

- static `size_t _S_nword (size_t __n)`

## Protected Attributes

- `_Bvector_impl _M_impl`

## Friends

- `template<typename >`  
`class hash`

## 4.1014.1 Detailed Description

`template<typename _Alloc> class std::vector< bool, _Alloc >`

A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.

## Template Parameters

|                     |                 |
|---------------------|-----------------|
| <code>_Alloc</code> | Allocator type. |
|---------------------|-----------------|

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

## See Also

`vector` for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 518 of file `stl_bvector.h`.

The documentation for this class was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

4.1015 `std::weak_ptr<_Tp>` Class Template Reference

Inherits `std::__weak_ptr<_Tp, _Lp>`.

## Public Types

- `typedef _Tp element_type`

## Public Member Functions

- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`  
`weak_ptr (const weak\_ptr<_Tp1 > &__r) noexcept`

- template<typename \_Tp1 , typename = typename std::enable\_if<std::is\_convertible<\_Tp1\*, \_Tp\*>::value>::type>  
**weak\_ptr** (const [shared\\_ptr](#)< \_Tp1 > &\_\_r) noexcept
- bool **expired** () const noexcept
- [shared\\_ptr](#)< \_Tp > **lock** () const noexcept
- template<typename \_Tp1 >  
[weak\\_ptr](#) & **operator=** (const [weak\\_ptr](#)< \_Tp1 > &\_\_r) noexcept
- template<typename \_Tp1 >  
[weak\\_ptr](#) & **operator=** (const [shared\\_ptr](#)< \_Tp1 > &\_\_r) noexcept
- template<typename \_Tp1 >  
bool **owner\_before** (const \_\_shared\_ptr< \_Tp1, \_Lp > &\_\_rhs) const
- template<typename \_Tp1 >  
bool **owner\_before** (const \_\_weak\_ptr< \_Tp1, \_Lp > &\_\_rhs) const
- void **reset** () noexcept
- void **swap** (\_\_weak\_ptr &\_\_s) noexcept
- long **use\_count** () const noexcept

#### 4.1015.1 Detailed Description

template<typename \_Tp>class std::weak\_ptr< \_Tp >

A smart pointer with weak semantics.

With forwarding constructors and assignment operators.

Definition at line 461 of file shared\_ptr.h.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 4.1016 std::weibull\_distribution< \_RealType > Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **weibull\_distribution** (\_RealType \_\_a=\_RealType(1), \_RealType \_\_b=\_RealType(1))
- **weibull\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator , typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator , typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)

- `_RealType a () const`
- `_RealType b () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `bool operator== (const weibull_distribution &__d1, const weibull_distribution &__d2)`

#### 4.1016.1 Detailed Description

`template<typename _RealType = double> class std::weibull_distribution<_RealType>`

A weibull\_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$$

Definition at line 4847 of file random.h.

#### 4.1016.2 Member Typedef Documentation

**4.1016.2.1** `template<typename _RealType = double> typedef _RealType std::weibull_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4850 of file random.h.

#### 4.1016.3 Member Function Documentation

**4.1016.3.1** `template<typename _RealType = double> _RealType std::weibull_distribution<_RealType>::a ( ) const`  
[inline]

Return the *a* parameter of the distribution.

Definition at line 4905 of file random.h.

**4.1016.3.2** `template<typename _RealType = double> _RealType std::weibull_distribution<_RealType>::b ( ) const`  
[inline]

Return the *b* parameter of the distribution.

Definition at line 4912 of file random.h.

4.1016.3.3 `template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4941 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

4.1016.3.4 `template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::min ( ) const`  
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4934 of file random.h.

4.1016.3.5 `template<typename _RealType = double> template<typename UniformRandomNumberGenerator > result_type`  
`std::weibull_distribution< _RealType >::operator() ( UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4949 of file random.h.

References `std::weibull_distribution< _RealType >::operator()()`.

Referenced by `std::weibull_distribution< _RealType >::operator()()`.

4.1016.3.6 `template<typename _RealType = double> param_type std::weibull_distribution< _RealType >::param ( ) const`  
`[inline]`

Returns the parameter set of the distribution.

Definition at line 4919 of file random.h.

Referenced by `std::operator>>()`.

4.1016.3.7 `template<typename _RealType = double> void std::weibull_distribution< _RealType >::param ( const`  
`param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4927 of file random.h.

4.1016.3.8 `template<typename _RealType = double> void std::weibull_distribution< _RealType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4898 of file random.h.

#### 4.1016.4 Friends And Related Function Documentation

4.1016.4.1 `template<typename _RealType = double> bool operator== ( const weibull_distribution< _RealType > & __d1,`  
`const weibull_distribution< _RealType > & __d2 ) [friend]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4984 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 4.1017 `std::weibull_distribution< _RealType >::param_type` Struct Reference

##### Public Types

- typedef [weibull\\_distribution](#)  
    < \_RealType > **distribution\_type**

##### Public Member Functions

- **param\_type** ( \_RealType \_\_a=\_RealType(1), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

##### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

##### 4.1017.1 Detailed Description

`template<typename _RealType = double>struct std::weibull_distribution< _RealType >::param_type`

Parameter type.

Definition at line 4856 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.1018 `tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

##### 4.1018.1 Detailed Description

`template<typename Node_Update, bool _BTp>struct tree_metadata_helper< Node_Update, _BTp >`

Tree metadata helper.

Definition at line 58 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 4.1019 tree\_traits< Key, Data, Cmp\_Fn, Node\_Update, Tag, \_Alloc > Struct Template Reference

### 4.1019.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_,
typename _Alloc > class Node_Update, typename Tag, typename _Alloc>struct tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag,
_Alloc >
```

Tree traits class, primary template.

Definition at line 70 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 4.1020 trie\_metadata\_helper< Node\_Update, \_BTp > Struct Template Reference

### 4.1020.1 Detailed Description

```
template<typename Node_Update, bool _BTp>struct trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

Definition at line 58 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

## 4.1021 trie\_traits< Key, Data, \_ATraits, Node\_Update, Tag, \_Alloc > Struct Template Reference

### 4.1021.1 Detailed Description

```
template<typename Key, typename Data, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_,
typename _Alloc > class Node_Update, typename Tag, typename _Alloc>struct trie_traits< Key, Data, _ATraits, Node_Update, Tag,
_Alloc >
```

Trie traits class, primary template.

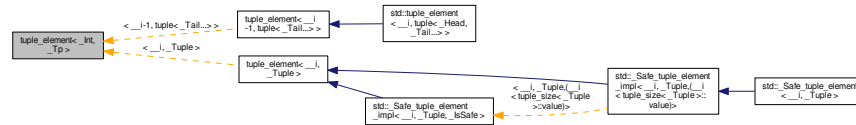
Definition at line 83 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 4.1022 tuple\_element< \_Int, \_Tp > Class Template Reference

Inheritance diagram for tuple\_element< \_Int, \_Tp >:



### 4.1022.1 Detailed Description

```
template<std::size_t _Int, typename _Tp>class tuple_element< _Int, _Tp >
```

tuple\_element

Definition at line 311 of file array.

The documentation for this class was generated from the following file:

- [array](#)

## 4.1023 tuple\_element< \_\_i, \_Tp > Struct Template Reference

### 4.1023.1 Detailed Description

```
template<std::size_t __i, typename _Tp>struct tuple_element< __i, _Tp >
```

Gives the type of the ith element of a given tuple type.

Definition at line 673 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.1024 tuple\_size< \_Tp > Class Template Reference

### 4.1024.1 Detailed Description

```
template<typename _Tp>class tuple_size< _Tp >
```

tuple\_size

Definition at line 303 of file array.

The documentation for this class was generated from the following file:

- [array](#)

## 4.1025 tuple\_size< \_Tp > Struct Template Reference

### 4.1025.1 Detailed Description

template<typename \_Tp>struct tuple\_size< \_Tp >

Finds the size of a given tuple type.

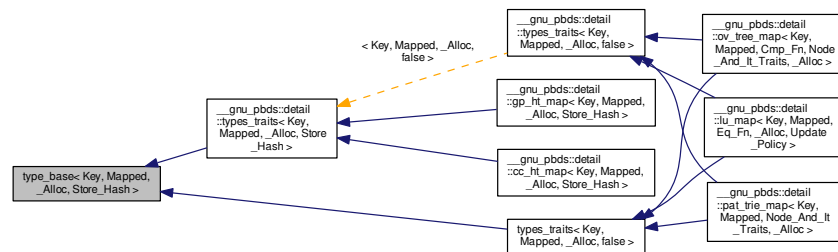
Definition at line 715 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.1026 type\_base< Key, Mapped, \_Alloc, Store\_Hash > Struct Template Reference

Inheritance diagram for type\_base< Key, Mapped, \_Alloc, Store\_Hash >:



### 4.1026.1 Detailed Description

template<typename Key, typename Mapped, typename \_Alloc, bool Store\_Hash>struct type\_base< Key, Mapped, \_Alloc, Store\_Hash >

Primary template.

Definition at line 107 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.1027 uses\_allocator< typename, typename > Struct Template Reference

### 4.1027.1 Detailed Description

template<typename, typename>struct uses\_allocator< typename, typename >

Declare uses\_allocator so it can be specialized in <queue> etc.

Definition at line 155 of file allocator.h.

The documentation for this struct was generated from the following file:

- [allocator.h](#)

## 5 File Documentation

### 5.1 algo.h File Reference

#### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber<\\_MustBeInt>](#)  
*Functor wrapper for std::rand().*

#### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

#### Functions

- template<typename \_RAIter >  
\_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, random\_access\_iterator\_tag)
- template<typename \_FIterator, typename \_IteratorTag >  
\_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_IteratorTag)
- template<typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag >  
\_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_BinaryPredicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_BinaryPredicate >  
\_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_BinaryPredicate \_\_pred, random\_access\_iterator\_tag)
- template<typename \_RAIter, typename \_Predicate >  
iterator\_traits<\_RAIter >  
::difference\_type **std::\_\_parallel::\_\_count\_if\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag=[\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_Iter, typename \_Predicate, typename \_IteratorTag >  
iterator\_traits<\_Iter >  
::difference\_type **std::\_\_parallel::\_\_count\_if\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Predicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_Tp >  
iterator\_traits<\_RAIter >  
::difference\_type **std::\_\_parallel::\_\_count\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, const \_Tp &\_\_value, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag=[\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_Iter, typename \_Tp, typename \_IteratorTag >  
iterator\_traits<\_Iter >  
::difference\_type **std::\_\_parallel::\_\_count\_switch** (\_Iter \_\_begin, \_Iter \_\_end, const \_Tp &\_\_value, \_IteratorTag)
- template<typename \_Iter, typename \_FIterator, typename \_IteratorTag1, typename \_IteratorTag2 >  
\_Iter **std::\_\_parallel::\_\_find\_first\_of\_switch** (\_Iter \_\_begin1, \_Iter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, \_IteratorTag1, \_IteratorTag2)

- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_RAIter std::parallel::find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _IIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_IIter std::parallel::find_first_of_switch (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter, typename _Predicate, typename _IteratorTag >`  
`_IIter std::parallel::find_if_switch (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _IIter, typename _Tp, typename _IteratorTag >`  
`_IIter std::parallel::find_switch (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _IIter, typename _Function, typename _IteratorTag >`  
`_Function std::parallel::for_each_switch (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`  
`_OutputIterator std::parallel::generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`  
`void std::parallel::generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::parallel::merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`

- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator std::parallel::partition_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, _`  
`IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_`  
`access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void std::parallel::replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp`  
`& __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _`  
`Tp & __new_value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void std::parallel::replace_switch (_FIterator __begin, _FIterator __end, const _Tp & __old_value, const`  
`_Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _`  
`Tp & __new_value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter std::parallel::search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &`  
`__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator std::parallel::search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const`  
`_Tp & __val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2,`  
`_FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2,`  
`_FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _`  
`IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __`  
`begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_`  
`access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _`  
`IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2`  
`__begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_`  
`access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`  
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`  
`RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type std::__parallel::count ( _Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type std::__parallel::count ( _Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen)`

- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end)`

- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`

### 5.1.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algo.h](#).

## 5.2 `algbase.h` File Reference

### Namespaces

- namespace `std`
- namespace `std::__parallel`

### Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool std::__parallel::__lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::__parallel::__lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::__mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::__parallel::__mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`

## 5.2.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algobase.h](#).

## 5.3 algorithm File Reference

## Macros

- `#define _GLIBCXX_ALGORITHM`

## 5.3.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [algorithm](#).

## 5.4 algorithm File Reference

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_ALGORITHM`

## Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator,`  
`_OutputIterator > __gnu_cxx::__copy_n (_InputIterator __first, _Size __count, _OutputIterator __result, input_`  
`iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`  
`pair< _RAIterator,`  
`_OutputIterator > __gnu_cxx::__copy_n (_RAIterator __first, _Size __count, _OutputIterator __result, random_`  
`_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _Input`  
`Iterator2 __first2, _InputIterator2 __last2)`
- `int __gnu_cxx::__lexicographical_compare_3way (const unsigned char *__first1, const unsigned char *__`  
`last1, const unsigned char *__first2, const unsigned char *__last2)`
- `int __gnu_cxx::__lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__`  
`first2, const char *__last2)`
- `template<typename _Tp >`  
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`

- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random-`  
`AccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random-`  
`AccessIterator __out, _RandomNumberGenerator & __rand, const _Distance __n)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator,`  
`_OutputIterator > gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`  
`void gnu_cxx::count (_InputIterator __first, _InputIterator __last, const _Tp & __value, _Size & __n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`  
`void gnu_cxx::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size & __n)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random-`  
`AccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random-`  
`AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator & __rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__out, const _Distance __n, _RandomNumberGenerator & __rand)`

#### 5.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/algorithm](#).

## 5.5 algorithm File Reference

### Macros

- `#define _PARALLEL_ALGORITHM`

#### 5.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [parallel/algorithm](#).

## 5.6 algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Filter >`  
`_Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2 >`  
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`  
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::count_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if (_Iter, _Iter, _Predicate)`

- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`  
`_Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`  
`void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`  
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`bool std::is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _Filter >`  
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`  
`void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`  
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::make_heap (_RAIter, _RAIter, _Compare)`

- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _BIter >`  
`bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::none_of (_Iter, _Iter, _Predicate)`
- `template<typename _RAIter >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter)`

- `template<typename _RAIter, typename _Compare >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`  
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`  
`pair< _OIter1, _OIter2 > std::partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter >`  
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter >`  
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`  
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`_OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp >`  
`void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`_OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter std::replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _BIter >`  
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`  
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`

- `template<typename _Filter >`  
`void std::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _OIter >`  
`_OIter std::rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`  
`_Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`_Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAIter, typename _UGenerator >`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter >`  
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_move_constructible< _Tp >`
- `template<typename _Tp, size_t _Nm>`  
`void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >  
_OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Filter >  
_Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >  
_Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >  
_OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >  
_OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp >  
_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >  
_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

#### Variables

- `void * std::__b`

#### 5.6.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

Definition in file <bits/algorithmfwd.h>.

## 5.7 algorithmfwd.h File Reference

#### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

#### Functions

- `template<typename _Filter, typename _IterTag >  
_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >  
_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >  
_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter >  
_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >  
iterator_traits< _Iter >  
::difference_type std::__parallel::__count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >  
iterator_traits< _RAIter >  
::difference_type std::__parallel::__count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`

- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >`  
`::difference_type std::parallel::count_switch ( _RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::find_first_of_switch ( _Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_RAIter std::parallel::find_first_of_switch ( _RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::find_first_of_switch ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter std::parallel::find_if_switch ( _Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch ( _RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter std::parallel::find_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::for_each_switch ( _RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function std::parallel::for_each_switch ( _Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter std::parallel::generate_n_switch ( _OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch ( _RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`  
`void std::parallel::generate_switch ( _Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch ( _RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool std::parallel::lexicographical_compare_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::parallel::max_element_switch ( _Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::max_element_switch ( _RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`  
`_Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`  
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`  
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`  
`_RandomAccess_OIter std::parallel::unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`

- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >`  
`::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`  
`_Iter std::parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter std::parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter std::parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`  
`_Iter std::parallel::find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::parallel::for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`

- `template<typename _Iterator, typename _Function >`  
`_Function std::parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void std::parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void std::parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void std::parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter std::parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter >`  
`_Filter std::parallel::min_element (_Filter, _Filter)`

- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _IIter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_IIter, _IIter, _OIter, _UnaryOperation)`
- `template<typename _IIter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_IIter, _IIter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_IIter, _IIter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_IIter, _IIter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_IIter, _IIter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_IIter, _IIter, _OIter)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_IIter, _IIter, _OIter, _Predicate)`

### 5.7.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/algorithmfwd.h](#).

## 5.8 alloc\_traits.h File Reference

### Classes

- struct [std::allocator\\_traits<\\_Alloc>](#)  
*Uniform interface to all allocator types.*

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_ALLOC_TR_NESTED_TYPE(_NTYPE, _ALT)`

### Typedefs

- `template<typename _Alloc>`  
`using std::__check_copy_constructible = __allow_copy_cons<__is_copy_insertable<_Alloc>::value>`

### Functions

- `template<typename _Alloc>`  
`void std::__alloc_on_copy (_Alloc &__one, const _Alloc &__two)`
- `template<typename _Alloc>`  
`_Alloc std::__alloc_on_copy (const _Alloc &__a)`
- `template<typename _Alloc>`  
`void std::__alloc_on_move (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc>`  
`void std::__alloc_on_swap (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc>`  
`void std::__do_alloc_on_copy (_Alloc &__one, const _Alloc &__two, true_type)`
- `template<typename _Alloc>`  
`void std::__do_alloc_on_move (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc>`  
`void std::__do_alloc_on_swap (_Alloc &__one, _Alloc &__two, true_type)`

### 5.8.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [bits/alloc\\_traits.h](#).

## 5.9 alloc\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits<\\_Alloc>](#)  
*Uniform interface to C++98 and C++0x allocators.*
- struct [std::allocator<\\_Tp>](#)  
*The standard allocator, as per [20.4].*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

#### 5.9.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [ext/alloc\\_traits.h](#).

## 5.10 allocator.h File Reference

### Classes

- struct [std::allocator<\\_Tp>](#)  
*The standard allocator, as per [20.4].*
- class [std::allocator<void>](#)  
*allocator<void> specialization.*
- struct [uses\\_allocator<typename, typename>](#)  
*Declare uses\_allocator so it can be specialized in <queue> etc.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _T1, typename _T2>  
bool std::operator!= (const allocator<_T1> &, const allocator<_T2> &)`
- `template<typename _Tp>  
bool std::operator!= (const allocator<_Tp> &, const allocator<_Tp> &)`
- `template<typename _T1, typename _T2>  
bool std::operator== (const allocator<_T1> &, const allocator<_T2> &)`
- `template<typename _Tp>  
bool std::operator== (const allocator<_Tp> &, const allocator<_Tp> &)`

## 5.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [allocator.h](#).

## 5.11 array File Reference

## Classes

- struct [std::array<\\_Tp, \\_Nm>](#)  
*A standard container for storing a fixed size sequence of elements.*
- class [tuple\\_element<\\_Int, \\_Tp>](#)  
*tuple\_element*
- class [tuple\\_size<\\_Tp>](#)  
*tuple\_size*

## Namespaces

- namespace [std](#)

## Macros

- `#define \_GLIBCXX\_ARRAY`

## Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & std::get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && std::get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & std::get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
void std::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`

## 5.11.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [array](#).

5.12 `array_allocator.h` File Reference

## Classes

- class [\\_\\_gnu\\_cxx::array\\_allocator< \\_Tp, \\_Array >](#)  
*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class [\\_\\_gnu\\_cxx::array\\_allocator\\_base< \\_Tp >](#)  
*Base class.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp, typename _Array >`  
`bool \_\_gnu\_cxx::operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp, typename _Array >`  
`bool \_\_gnu\_cxx::operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

## 5.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [array\\_allocator.h](#).

5.13 `assoc_container.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::basic\\_branch< Key, Mapped, Tag, Node\\_Update, Policy\\_Tl, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::basic\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Resize\\_Policy, Store\\_Hash, Tag, Policy\\_Tl, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::cc\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Hash\\_Fn, Resize\\_Policy, Store\\_Hash, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::gp\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Probe\\_Fn, Probe\\_Fn, Resize\\_Policy, Store\\_Hash, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::list\\_update< Key, Mapped, Eq\\_Fn, Update\\_Policy, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::tree< Key, Mapped, Cmp\\_Fn, Tag, Node\\_Update, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::trie< Key, Mapped, \\_ATraits, Tag, Node\\_Update, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`
- `#define PB_DS_LU_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

## 5.13.1 Detailed Description

Contains associative containers.

Definition in file [assoc\\_container.hpp](#).

## 5.14 atomic File Reference

## Classes

- struct [std::atomic< \\_Tp >](#)  
*Generic atomic type, primary class template.*
- struct [std::atomic< \\_Tp \\* >](#)  
*Partial specialization for pointer types.*
- struct [std::atomic< bool >](#)  
*Explicit specialization for bool.*
- struct [std::atomic< char >](#)  
*Explicit specialization for char.*
- struct [std::atomic< char16\\_t >](#)  
*Explicit specialization for char16\_t.*
- struct [std::atomic< char32\\_t >](#)  
*Explicit specialization for char32\_t.*
- struct [std::atomic< int >](#)  
*Explicit specialization for int.*
- struct [std::atomic< long >](#)  
*Explicit specialization for long.*
- struct [std::atomic< long long >](#)  
*Explicit specialization for long long.*
- struct [std::atomic< short >](#)  
*Explicit specialization for short.*
- struct [std::atomic< signed char >](#)  
*Explicit specialization for signed char.*
- struct [std::atomic< unsigned char >](#)  
*Explicit specialization for unsigned char.*
- struct [std::atomic< unsigned int >](#)  
*Explicit specialization for unsigned int.*

- struct `std::atomic< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `std::atomic< unsigned long long >`  
*Explicit specialization for unsigned long long.*
- struct `std::atomic< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*
- struct `std::atomic_bool`  
*atomic\_bool*

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_ATOMIC`

### Functions

- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`

- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub_explicit (atomic< _ITp > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_init (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_init (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`

#### 5.14.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic](#).

## 5.15 atomic\_base.h File Reference

### Classes

- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)  
*Base class for atomic integrals.*
- struct [std::\\_\\_atomic\\_base< \\_PTp \\* >](#)  
*Partial specialization for pointer types.*
- struct [std::\\_\\_atomic\\_flag\\_base](#)  
*Base type for atomic\_flag.*
- struct [std::atomic< \\_Tp >](#)  
*Generic atomic type, primary class template.*
- struct [std::atomic\\_flag](#)  
*atomic\_flag*

### Namespaces

- namespace [std](#)

### Macros

- #define **ATOMIC\_FLAG\_INIT**
- #define **ATOMIC\_VAR\_INIT**(\_VI)

### Typedefs

- typedef unsigned char **std::\_\_atomic\_flag\_data\_type**
- typedef \_\_atomic\_base< char > [std::atomic\\_char](#)
- typedef \_\_atomic\_base< char16\_t > [std::atomic\\_char16\\_t](#)
- typedef \_\_atomic\_base< char32\_t > [std::atomic\\_char32\\_t](#)
- typedef \_\_atomic\_base< int > [std::atomic\\_int](#)
- typedef \_\_atomic\_base< int\_fast16\_t > [std::atomic\\_int\\_fast16\\_t](#)
- typedef \_\_atomic\_base< int\_fast32\_t > [std::atomic\\_int\\_fast32\\_t](#)
- typedef \_\_atomic\_base< int\_fast64\_t > [std::atomic\\_int\\_fast64\\_t](#)
- typedef \_\_atomic\_base< int\_fast8\_t > [std::atomic\\_int\\_fast8\\_t](#)
- typedef \_\_atomic\_base< int\_least16\_t > [std::atomic\\_int\\_least16\\_t](#)
- typedef \_\_atomic\_base< int\_least32\_t > [std::atomic\\_int\\_least32\\_t](#)
- typedef \_\_atomic\_base< int\_least64\_t > [std::atomic\\_int\\_least64\\_t](#)
- typedef \_\_atomic\_base< int\_least8\_t > [std::atomic\\_int\\_least8\\_t](#)
- typedef \_\_atomic\_base< intmax\_t > [std::atomic\\_intmax\\_t](#)
- typedef \_\_atomic\_base< intptr\_t > [std::atomic\\_intptr\\_t](#)
- typedef \_\_atomic\_base< long long > [std::atomic\\_llong](#)

- typedef \_\_atomic\_base< long > [std::atomic\\_long](#)
- typedef \_\_atomic\_base< ptrdiff\_t > [std::atomic\\_ptrdiff\\_t](#)
- typedef \_\_atomic\_base< signed char > [std::atomic\\_schar](#)
- typedef \_\_atomic\_base< short > [std::atomic\\_short](#)
- typedef \_\_atomic\_base< size\_t > [std::atomic\\_size\\_t](#)
- typedef \_\_atomic\_base< unsigned char > [std::atomic\\_uchar](#)
- typedef \_\_atomic\_base< unsigned int > [std::atomic\\_uint](#)
- typedef \_\_atomic\_base< uint\_fast16\_t > [std::atomic\\_uint\\_fast16\\_t](#)
- typedef \_\_atomic\_base< uint\_fast32\_t > [std::atomic\\_uint\\_fast32\\_t](#)
- typedef \_\_atomic\_base< uint\_fast64\_t > [std::atomic\\_uint\\_fast64\\_t](#)
- typedef \_\_atomic\_base< uint\_fast8\_t > [std::atomic\\_uint\\_fast8\\_t](#)
- typedef \_\_atomic\_base< uint\_least16\_t > [std::atomic\\_uint\\_least16\\_t](#)
- typedef \_\_atomic\_base< uint\_least32\_t > [std::atomic\\_uint\\_least32\\_t](#)
- typedef \_\_atomic\_base< uint\_least64\_t > [std::atomic\\_uint\\_least64\\_t](#)
- typedef \_\_atomic\_base< uint\_least8\_t > [std::atomic\\_uint\\_least8\\_t](#)
- typedef \_\_atomic\_base< uintmax\_t > [std::atomic\\_uintmax\\_t](#)
- typedef \_\_atomic\_base< uintptr\_t > [std::atomic\\_uintptr\\_t](#)
- typedef \_\_atomic\_base< unsigned long long > [std::atomic\\_ullong](#)
- typedef \_\_atomic\_base< unsigned long > [std::atomic\\_ulong](#)
- typedef \_\_atomic\_base< unsigned short > [std::atomic\\_ushort](#)
- typedef \_\_atomic\_base< wchar\_t > [std::atomic\\_wchar\\_t](#)
- typedef enum [std::memory\\_order](#) [std::memory\\_order](#)

#### Enumerations

- enum \_\_memory\_order\_modifier { \_\_memory\_order\_mask, \_\_memory\_order\_modifier\_mask, \_\_memory\_order\_hle\_acquire, \_\_memory\_order\_hle\_release }
- enum [std::memory\\_order](#) { [memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#), [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

#### Functions

- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order](#) (memory\_order \_\_m) noexcept
- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order2](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_signal\\_fence](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_thread\\_fence](#) (memory\_order \_\_m) noexcept

- `template<typename _Tp >`  
`_Tp std::kill\_dependency (_Tp __y) noexcept`
- `constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod)`
- `constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod)`

#### 5.15.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

Definition in file [atomic\\_base.h](#).

## 5.16 `atomic_lockfree_defines.h` File Reference

### Macros

- `#define ATOMIC\_BOOL\_LOCK\_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

#### 5.16.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

Definition in file [atomic\\_lockfree\\_defines.h](#).

## 5.17 `atomic_word.h` File Reference

### Typedefs

- `typedef int _Atomic_word`

#### 5.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomic\\_word.h](#).

## 5.18 `atomicity.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _GLIBCXX_READ_MEM_BARRIER`
- `#define _GLIBCXX_WRITE_MEM_BARRIER`

## Functions

- `static void __gnu_cxx::__atomic_add_single (_Atomic_word * __mem, int __val)`
- `else __gnu_cxx::__atomic_add_single (__mem, __val)`
- `_Atomic_word __gnu_cxx::__attribute__((__unused__)) __exchange_and_add(volatile _Atomic_word *`
- `static _Atomic_word __gnu_cxx::__exchange_and_add_single (_Atomic_word * __mem, int __val)`
- `else return __gnu_cxx::__exchange_and_add_single (__mem, __val)`
- `static _Atomic_word int __val __gnu_cxx::if (__gthread_active_p()) return __exchange_and_add(__mem`
- `_Atomic_word int __gnu_cxx::throw ()`

## Variables

- `static _Atomic_word int __val __gnu_cxx::__val`

## 5.18.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomicity.h](#).

## 5.19 auto\_ptr.h File Reference

## Classes

- class [std::auto\\_ptr<\\_Tp>](#)  
*A simple smart pointer providing strict ownership semantics.*
- struct [std::auto\\_ptr\\_ref<\\_Tp1>](#)

## Namespaces

- namespace [std](#)

## 5.19.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [auto\\_ptr.h](#).

## 5.20 backward\_warning.h File Reference

### 5.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [backward\\_warning.h](#).

## 5.21 balanced\_quicksort.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>](#)  
*Information local to one thread in the parallel quicksort run.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer](#) (\_QSBThreadLocal<\_RAIter> \*\_\_tls, \_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_iam, \_ThreadIndex \_\_num\_threads, bool \_\_parent\_wait)
- template<typename \_RAIter, typename \_Compare>  
std::iterator\_traits<\_RAIter>  
::difference\_type [\\_\\_gnu\\_parallel::\\_\\_qsb\\_divide](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping](#) (\_QSBThreadLocal<\_RAIter> \*\_\_tls, \_Compare &\_\_comp, \_ThreadIndex \_\_iam, bool \_\_wait)

### 5.21.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort. It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [balanced\\_quicksort.h](#).

## 5.22 base.h File Reference

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)
- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## 5.22.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/base.h](#).

## 5.23 base.h File Reference

## Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#)  
*Similar to [std::binder1st](#), but giving the argument types explicitly.*
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#)  
*Similar to [std::binder2nd](#), but giving the argument types explicitly.*
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate<\\_Predicate, argument\\_type >](#)  
*Similar to [std::unary\\_negate](#), but giving the argument types explicitly.*
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess<\\_T1, \\_T2, \\_Compare >](#)  
*Constructs predicate for equality from strict weak ordering predicate.*
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo<\\_T1, \\_T2 >](#)  
*Similar to [std::equal\\_to](#), but allows two different types.*
- struct [\\_\\_gnu\\_parallel::\\_\\_Less<\\_T1, \\_T2 >](#)  
*Similar to [std::less](#), but allows two different types.*
- struct [\\_\\_gnu\\_parallel::\\_\\_Multiplies<\\_Tp1, \\_Tp2, \\_Result >](#)  
*Similar to [std::multiplies](#), but allows two different types.*
- struct [\\_\\_gnu\\_parallel::\\_\\_Plus<\\_Tp1, \\_Tp2, \\_Result >](#)  
*Similar to [std::plus](#), but allows two different types.*
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence<\\_Tp, \\_DifferenceTp >](#)  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequenceIterator<\\_Tp, \\_DifferenceTp >](#)  
*\_Iterator associated with [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence](#). It features the usual random-access iterator functionality.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)
- namespace [\\_\\_gnu\\_sequential](#)
- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_PARALLEL\_ASSERT(_Condition)`

## Functions

- `void __gnu_parallel::__decode2` (`_CASable __x`, `int &__a`, `int &__b`)
- `_CASable __gnu_parallel::__encode2` (`int __a`, `int __b`)
- `_ThreadIndex __gnu_parallel::__get_max_threads` ()
- `bool __gnu_parallel::__is_parallel` (`const _Parallelism __p`)
- `template<typename _RAlter, typename _Compare >`  
`_RAlter __gnu_parallel::__median_of_three_iterators` (`_RAlter __a`, `_RAlter __b`, `_RAlter __c`, `_Compare __-`  
`comp`)
- `template<typename _Size >`  
`_Size __gnu_parallel::__rd_log2` (`_Size __n`)
- `template<typename _Tp >`  
`const _Tp & __gnu_parallel::max` (`const _Tp &__a`, `const _Tp &__b`)
- `template<typename _Tp >`  
`const _Tp & __gnu_parallel::min` (`const _Tp &__a`, `const _Tp &__b`)

## 5.23.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/base.h](#).

5.24 `basic_file.h` File Reference

## Namespaces

- namespace [std](#)

## 5.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [basic\\_file.h](#).

5.25 `basic_ios.h` File Reference

## Classes

- class [std::basic\\_ios<\\_CharT, \\_Traits >](#)  
*Template class `basic_ios`, virtual base class for all stream classes.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >`  
`const _Facet & std::__check_facet` (`const _Facet *__f`)

### 5.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [basic\\_ios.h](#).

## 5.26 `basic_ios.tcc` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _BASIC_IOS_TCC`

### 5.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [basic\\_ios.tcc](#).

## 5.27 `basic_iterator.h` File Reference

### 5.27.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [basic\\_iterator.h](#).

## 5.28 `basic_string.h` File Reference

### Classes

- class [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>](#)  
*Managing sequences of characters and character-like objects.*
- struct [std::hash<string>](#)  
*std::hash specialization for string.*
- struct [std::hash<u16string>](#)  
*std::hash specialization for u16string.*
- struct [std::hash<u32string>](#)  
*std::hash specialization for u32string.*
- struct [std::hash<wstring>](#)  
*std::hash specialization for wstring.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<>`  
`basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > std::operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > std::operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits,`  
`_Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_`  
`string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if`  
`< __is_char< _CharT >::__value,`  
`bool >::__type std::operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > & __is, basic_string< char > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void std::swap (basic_string< _CharT, _Traits, _Alloc > & __lhs, basic_string< _CharT, _Traits, _Alloc > & __rhs)`

### 5.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [basic\\_string.h](#).

## 5.29 `basic_string.tcc` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define \_BASIC\_STRING\_TCC`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str)`

### 5.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [basic\\_string.tcc](#).

## 5.30 bin\_search\_tree\_.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

#### 5.30.1 Detailed Description

Contains an implementation class for binary search tree.

Definition in file [bin\\_search\\_tree\\_.hpp](#).

## 5.31 binary\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

#### 5.31.1 Detailed Description

Contains an implementation class for a binary heap.

Definition in file [binary\\_heap\\_.hpp](#).

## 5.32 binders.h File Reference

### Classes

- class [std::binder1st<\\_Operation>](#)   
*One of the [binder](#) functors.*
- class [std::binder2nd<\\_Operation>](#)   
*One of the [binder](#) functors.*

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Operation, typename \_Tp>   
[binder1st<\\_Operation>](#) [std::bind1st](#) (const \_Operation &\_\_fn, const \_Tp &\_\_x)
- template<typename \_Operation, typename \_Tp>   
[binder2nd<\\_Operation>](#) [std::bind2nd](#) (const \_Operation &\_\_fn, const \_Tp &\_\_x)

#### 5.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [binders.h](#).

## 5.33 binomial\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap<Value\\_Type, Cmp\\_Fn, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB\_DS\_CLASS\_C\_DEC`
- `#define PB\_DS\_CLASS\_T\_DEC`

#### 5.33.1 Detailed Description

Contains an implementation class for a binomial heap.

Definition in file [binomial\\_heap\\_.hpp](#).

## 5.34 binomial\_heap\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)  
*Base class for binomial heap.*

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_ASSERT\_BASE\_NODE\_CONSISTENT**(\_Node, \_Bool)
- #define **PB\_DS\_ASSERT\_VALID\_COND**(X, \_StrictlyBinomial)
- #define **PB\_DS\_B\_HEAP\_BASE**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

#### 5.34.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial\\_heap\\_base.hpp](#).

## 5.35 bitmap\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector< \\_Tp >](#)  
*[\\_\\_mini\\_vector<>](#) is a stripped down version of the full-fledged [std::vector<>](#).*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::Bitmap\\_counter< \\_Tp >](#)  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::Ffit\\_finder< \\_Tp >](#)  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp >](#)  
*Bitmap Allocator, primary template.*
- class [\\_\\_gnu\\_cxx::free\\_list](#)  
*The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)

### Macros

- #define [\\_BALLOC\\_ALIGN\\_BYTES](#)

## Enumerations

- enum { **bits\_per\_byte**, **bits\_per\_block** }

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)
- size\_t [\\_\\_gnu\\_cxx::\\_\\_Bit\\_scan\\_forward](#) (size\_t \_\_num)
- template<typename \_Tp1, typename \_Tp2 >  
bool [\\_\\_gnu\\_cxx::operator!=](#) (const bitmap\_allocator< \_Tp1 > &, const bitmap\_allocator< \_Tp2 > &) throw ()
- template<typename \_Tp1, typename \_Tp2 >  
bool [\\_\\_gnu\\_cxx::operator==](#) (const bitmap\_allocator< \_Tp1 > &, const bitmap\_allocator< \_Tp2 > &) throw ()

## 5.35.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [bitmap\\_allocator.h](#).

## 5.35.2 Macro Definition Documentation

## 5.35.2.1 #define \_BALLOC\_ALIGN\_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 43 of file [bitmap\\_allocator.h](#).

## 5.36 bitset File Reference

## Classes

- struct [std::\\_Base\\_bitset< \\_Nw >](#)
- struct [std::\\_Base\\_bitset< 0 >](#)
- struct [std::\\_Base\\_bitset< 1 >](#)
- struct [std::hash<::bitset< \\_Nb > >](#)  
*std::hash specialization for bitset.*

## Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_BITSET`
- `#define _GLIBCXX_BITSET_BITS_PER_ULL`
- `#define _GLIBCXX_BITSET_BITS_PER_WORD`
- `#define _GLIBCXX_BITSET_WORDS(__n)`

## Functions

- `size_t std::Find_first ()` const noexcept
- `size_t std::Find_next (size_t __prev)` const noexcept
- `template<class _CharT, class _Traits >`  
`void std::M_copy_from_ptr (const _CharT *, size_t, size_t, size_t, _CharT, _CharT)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void std::M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void std::M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void std::M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &, _CharT, _CharT) const`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void std::M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &__s) const`
- `template<size_t _Nb>`  
`std::M_do_and (__rhs)`
- `bool std::all ()` const noexcept
- `bool std::any ()` const noexcept
- `size_t std::count ()` const noexcept
- `bitset< _Nb > & std::flip ()` noexcept
- `bitset< _Nb > & std::flip (size_t __position)`
- `bool std::none ()` const noexcept
- `bitset< _Nb > & std::operator^= (const bitset< _Nb > &__rhs)` noexcept
- `bitset< _Nb > & std::operator|= (const bitset< _Nb > &__rhs)` noexcept
- `bitset< _Nb > std::operator~ ()` const noexcept
- `bitset< _Nb > & std::reset ()` noexcept
- `bitset< _Nb > & std::reset (size_t __position)`
- `bitset< _Nb > & std::set ()` noexcept
- `bitset< _Nb > & std::set (size_t __position, bool __val=true)`
- `constexpr size_t std::size ()` const noexcept
- `bool std::test (size_t __position)` const
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_string< _CharT, _Traits, _Alloc > std::to_string ()` const
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_string< _CharT, _Traits, _Alloc > std::to_string (_CharT __zero, _CharT __one=_CharT('1'))` const
- `template<class _CharT, class _Traits >`  
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > std::to_string ()` const

- `template<class _CharT, class _Traits >`  
`std::basic_string< _CharT,`  
`_Traits, std::allocator`  
`< _CharT > > std::to_string ( _CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT >`  
`std::basic_string< _CharT,`  
`std::char_traits< _CharT >`  
`, std::allocator< _CharT > > std::to_string () const`
- `template<class _CharT >`  
`std::basic_string< _CharT,`  
`std::char_traits< _CharT >`  
`, std::allocator< _CharT > > std::to_string ( _CharT __zero, _CharT __one=_CharT('1')) const`
- `std::basic_string< char,`  
`std::char_traits< char >`  
`, std::allocator< char > > std::to_string (char __zero, char __one= '1') const`
- unsigned long long `std::to_ullong () const`
- unsigned long `std::to_ulong () const`
  
- `bitset< _Nb > & std::operator<=<= (size_t __position) noexcept`
- `bitset< _Nb > & std::operator>=>= (size_t __position) noexcept`
  
- `bitset< _Nb > & std::_Unchecked_set (size_t __pos) noexcept`
- `bitset< _Nb > & std::_Unchecked_set (size_t __pos, int __val) noexcept`
- `bitset< _Nb > & std::_Unchecked_reset (size_t __pos) noexcept`
- `bitset< _Nb > & std::_Unchecked_flip (size_t __pos) noexcept`
- `constexpr bool std::_Unchecked_test (size_t __pos) const noexcept`
  
- reference `std::operator[] (size_t __position)`
  
- bool `std::operator== (const bitset< _Nb > &__rhs) const noexcept`
- bool `std::operator!= (const bitset< _Nb > &__rhs) const noexcept`
  
- `bitset< _Nb > std::operator<< (size_t __position) const noexcept`
- `bitset< _Nb > std::operator>> (size_t __position) const noexcept`
  
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
  
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream< _CharT,`  
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT,`  
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`

## Variables

- return \* `std::this`

## 5.36.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [bitset](#).

5.37 **bitset File Reference**

## Classes

- class [std::\\_\\_debug::bitset< \\_Nb >](#)  
*Class std::bitset with additional safety/checking/debug instrumentation.*
- struct [std::hash< \\_\\_debug::bitset< \\_Nb > >](#)  
*std::hash specialization for bitset.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<size_t _Nb>  
bitset< _Nb > std::\_\_debug::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_ostream< _CharT,  
_Traits > & std::\_\_debug::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb >  
&__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_istream< _CharT,  
_Traits > & std::\_\_debug::operator>> (std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>  
bitset< _Nb > std::\_\_debug::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>  
bitset< _Nb > std::\_\_debug::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`

## 5.37.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/bitset](#).

5.38 **bitset File Reference**

## Classes

- class [std::\\_\\_profile::bitset< \\_Nb >](#)  
*Class std::bitset wrapper with performance instrumentation.*
- struct [std::hash< \\_\\_profile::bitset< \\_Nb > >](#)  
*std::hash specialization for bitset.*

## Namespaces

- namespace `std`
- namespace `std::__profile`

## Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT,`  
`_Traits > & std::__profile::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb >`  
`&__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT,`  
`_Traits > & std::__profile::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`

## 5.38.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/bitset](#).

5.39 `bool_set` File Reference

## Classes

- class `std::tr2::bool_set`

## Namespaces

- namespace `std`
- namespace `std::tr2`

## Macros

- `#define _GLIBCXX_TR2_BOOL_SET`

## Functions

- `bool std::tr2::certainly (bool_set __b)`
- `bool std::tr2::contains (bool_set __s, bool_set __t)`
- `bool std::tr2::equals (bool_set __s, bool_set __t)`
- `bool std::tr2::is_emptyset (bool_set __b)`
- `bool std::tr2::is_indeterminate (bool_set __b)`
- `bool std::tr2::is_singleton (bool_set __b)`

- `bool_set std::tr2::operator!= (bool __s, bool_set __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool_set __s, bool __t)`
- `bool_set std::tr2::operator== (bool __s, bool_set __t)`
- `bool_set std::tr2::operator== (bool_set __s, bool __t)`
- `bool_set std::tr2::operator^ (bool __s, bool_set __t)`
- `bool_set std::tr2::operator^ (bool_set __s, bool __t)`
- `bool_set std::tr2::operator| (bool __s, bool_set __t)`
- `bool_set std::tr2::operator| (bool_set __s, bool __t)`
- `bool std::tr2::possibly (bool_set __b)`
- `bool_set std::tr2::set_complement (bool_set __b)`
- `bool_set std::tr2::set_intersection (bool __s, bool_set __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool_set __t)`
- `bool_set std::tr2::set_union (bool __s, bool_set __t)`
- `bool_set std::tr2::set_union (bool_set __s, bool __t)`
- `bool_set std::tr2::set_union (bool_set __s, bool_set __t)`

#### 5.39.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file [bool\\_set](#).

## 5.40 bool\_set.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_BOOL_SET_TCC`

#### 5.40.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file [bool\\_set.tcc](#).

## 5.41 boost\_concept\_check.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define __GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define __GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define __GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define __GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define __GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define __GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define __IsUnused`

## Functions

- `template<class _Tp >`  
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`  
`void __gnu_cxx::__function_requires ()`

## 5.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [boost\\_concept\\_check.h](#).

## 5.42 branch\_policy.hpp File Reference

## Classes

- `struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`  
*Primary template, base class for branch structure policies.*
- `struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >`  
*Specialization for const iterators.*

## Namespaces

- namespace `__gnu_pbds`

## 5.42.1 Detailed Description

Contains a base class for branch policies.

Definition in file [branch\\_policy.hpp](#).

## 5.43 `c++0x_warning.h` File Reference

### 5.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [c++0x\\_warning.h](#).

## 5.44 `c++allocator.h` File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Tp >`  
`using std::\_\_allocator\_base = \_\_gnu\_cxx::new\_allocator< _Tp >`

### 5.44.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [c++allocator.h](#).

## 5.45 `c++config.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define \_\_GLIBCXX\_\_`
- `#define \_\_glibcxx\_assert(_Condition)`
- `#define \_\_N(msgid)`
- `#define \_GLIBCXX\_ATOMIC\_BUILTINS`
- `#define \_GLIBCXX\_BEGIN\_EXTERN\_C`
- `#define \_GLIBCXX\_BEGIN\_NAMESPACE\_LDBL`
- `#define \_GLIBCXX\_BEGIN\_NAMESPACE\_VERSION`
- `#define \_GLIBCXX\_DEPRECATED`
- `#define \_GLIBCXX\_END\_EXTERN\_C`
- `#define \_GLIBCXX\_END\_NAMESPACE\_LDBL`
- `#define \_GLIBCXX\_END\_NAMESPACE\_VERSION`
- `#define \_GLIBCXX\_EXTERN\_TEMPLATE`
- `#define \_GLIBCXX\_FAST\_MATH`
- `#define \_GLIBCXX\_FULLY\_DYNAMIC\_STRING`
- `#define \_GLIBCXX\_HAS\_GTHREADS`

- `#define _GLIBCXX_HAVE_ACOSF`
- `#define _GLIBCXX_HAVE_ACOSL`
- `#define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- `#define _GLIBCXX_HAVE_ASINF`
- `#define _GLIBCXX_HAVE_ASINL`
- `#define _GLIBCXX_HAVE_AT_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_ATAN2F`
- `#define _GLIBCXX_HAVE_ATAN2L`
- `#define _GLIBCXX_HAVE_ATANF`
- `#define _GLIBCXX_HAVE_ATANL`
- `#define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- `#define _GLIBCXX_HAVE_CEILF`
- `#define _GLIBCXX_HAVE_CEILL`
- `#define _GLIBCXX_HAVE_COMPLEX_H`
- `#define _GLIBCXX_HAVE_COSF`
- `#define _GLIBCXX_HAVE_COSHF`
- `#define _GLIBCXX_HAVE_COSHL`
- `#define _GLIBCXX_HAVE_COSL`
- `#define _GLIBCXX_HAVE_DLFCN_H`
- `#define _GLIBCXX_HAVE_EBADMSG`
- `#define _GLIBCXX_HAVE_ECANCELED`
- `#define _GLIBCXX_HAVE_ECHILD`
- `#define _GLIBCXX_HAVE_EIDRM`
- `#define _GLIBCXX_HAVE_ENDIAN_H`
- `#define _GLIBCXX_HAVE_ENODATA`
- `#define _GLIBCXX_HAVE_ENOLINK`
- `#define _GLIBCXX_HAVE_ENOSPC`
- `#define _GLIBCXX_HAVE_ENOSR`
- `#define _GLIBCXX_HAVE_ENOSTR`
- `#define _GLIBCXX_HAVE_ENOTRECOVERABLE`
- `#define _GLIBCXX_HAVE_ENOTSUP`
- `#define _GLIBCXX_HAVE_EOVERFLOW`
- `#define _GLIBCXX_HAVE_EOWNERDEAD`
- `#define _GLIBCXX_HAVE_EPERM`
- `#define _GLIBCXX_HAVE_EPROTO`
- `#define _GLIBCXX_HAVE_ETIME`
- `#define _GLIBCXX_HAVE_ETIMEDOUT`
- `#define _GLIBCXX_HAVE_ETXTBSY`
- `#define _GLIBCXX_HAVE_EWOULDBLOCK`
- `#define _GLIBCXX_HAVE_EXECINFO_H`
- `#define _GLIBCXX_HAVE_EXPF`
- `#define _GLIBCXX_HAVE_EXPL`
- `#define _GLIBCXX_HAVE_FABSF`
- `#define _GLIBCXX_HAVE_FABSL`
- `#define _GLIBCXX_HAVE_FENV_H`
- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOAT_H`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`

- #define \_GLIBCXX\_HAVE\_FMODF
- #define \_GLIBCXX\_HAVE\_FMODL
- #define \_GLIBCXX\_HAVE\_FREXPF
- #define \_GLIBCXX\_HAVE\_FREXPL
- #define \_GLIBCXX\_HAVE\_GETIPINFO
- #define \_GLIBCXX\_HAVE\_GETS
- #define \_GLIBCXX\_HAVE\_HYPOT
- #define \_GLIBCXX\_HAVE\_HYPOTF
- #define \_GLIBCXX\_HAVE\_HYPOTL
- #define \_GLIBCXX\_HAVE\_ICONV
- #define \_GLIBCXX\_HAVE\_INT64\_T
- #define \_GLIBCXX\_HAVE\_INT64\_T\_LONG
- #define \_GLIBCXX\_HAVE\_INTPTR\_T
- #define \_GLIBCXX\_HAVE\_ISINF
- #define \_GLIBCXX\_HAVE\_ISINFF
- #define \_GLIBCXX\_HAVE\_ISINFL
- #define \_GLIBCXX\_HAVE\_ISNAN
- #define \_GLIBCXX\_HAVE\_ISNANF
- #define \_GLIBCXX\_HAVE\_ISNANL
- #define \_GLIBCXX\_HAVE\_ISWBLANK
- #define \_GLIBCXX\_HAVE\_LC\_MESSAGES
- #define \_GLIBCXX\_HAVE\_LDEXPF
- #define \_GLIBCXX\_HAVE\_LDEXPL
- #define \_GLIBCXX\_HAVE\_LIBINTL\_H
- #define \_GLIBCXX\_HAVE\_LIMIT\_AS
- #define \_GLIBCXX\_HAVE\_LIMIT\_DATA
- #define \_GLIBCXX\_HAVE\_LIMIT\_FSIZE
- #define \_GLIBCXX\_HAVE\_LIMIT\_RSS
- #define \_GLIBCXX\_HAVE\_LIMIT\_VMEM
- #define \_GLIBCXX\_HAVE\_LINUX\_FUTEX
- #define \_GLIBCXX\_HAVE\_LOCALE\_H
- #define \_GLIBCXX\_HAVE\_LOG10F
- #define \_GLIBCXX\_HAVE\_LOG10L
- #define \_GLIBCXX\_HAVE\_LOGF
- #define \_GLIBCXX\_HAVE\_LOGL
- #define \_GLIBCXX\_HAVE\_MBSTATE\_T
- #define \_GLIBCXX\_HAVE\_MEMORY\_H
- #define \_GLIBCXX\_HAVE\_MODF
- #define \_GLIBCXX\_HAVE\_MODFF
- #define \_GLIBCXX\_HAVE\_MODFL
- #define \_GLIBCXX\_HAVE\_POLL
- #define \_GLIBCXX\_HAVE\_POWF
- #define \_GLIBCXX\_HAVE\_POWL
- #define \_GLIBCXX\_HAVE\_QUICK\_EXIT
- #define \_GLIBCXX\_HAVE\_S\_ISREG
- #define \_GLIBCXX\_HAVE\_SETENV
- #define \_GLIBCXX\_HAVE\_SINCOS
- #define \_GLIBCXX\_HAVE\_SINCOSF
- #define \_GLIBCXX\_HAVE\_SINCOSL
- #define \_GLIBCXX\_HAVE\_SINF
- #define \_GLIBCXX\_HAVE\_SINHF

- #define \_GLIBCXX\_HAVE\_SINHL
- #define \_GLIBCXX\_HAVE\_SINL
- #define \_GLIBCXX\_HAVE\_SLEEP
- #define \_GLIBCXX\_HAVE\_SQRTF
- #define \_GLIBCXX\_HAVE\_SQRTL
- #define \_GLIBCXX\_HAVE\_STDALIGN\_H
- #define \_GLIBCXX\_HAVE\_STDBOOL\_H
- #define \_GLIBCXX\_HAVE\_STDINT\_H
- #define \_GLIBCXX\_HAVE\_STDLIB\_H
- #define \_GLIBCXX\_HAVE\_STRERROR\_L
- #define \_GLIBCXX\_HAVE\_STRERROR\_R
- #define \_GLIBCXX\_HAVE\_STRING\_H
- #define \_GLIBCXX\_HAVE\_STRINGS\_H
- #define \_GLIBCXX\_HAVE\_STRTOF
- #define \_GLIBCXX\_HAVE\_STRTOLD
- #define \_GLIBCXX\_HAVE\_STRXFRM\_L
- #define \_GLIBCXX\_HAVE\_SYMVER\_SYMBOL\_RENAMING\_RUNTIME\_SUPPORT
- #define \_GLIBCXX\_HAVE\_SYS\_IOCTL\_H
- #define \_GLIBCXX\_HAVE\_SYS\_IPC\_H
- #define \_GLIBCXX\_HAVE\_SYS\_PARAM\_H
- #define \_GLIBCXX\_HAVE\_SYS\_RESOURCE\_H
- #define \_GLIBCXX\_HAVE\_SYS\_SEM\_H
- #define \_GLIBCXX\_HAVE\_SYS\_STAT\_H
- #define \_GLIBCXX\_HAVE\_SYS\_SYSINFO\_H
- #define \_GLIBCXX\_HAVE\_SYS\_TIME\_H
- #define \_GLIBCXX\_HAVE\_SYS\_TYPES\_H
- #define \_GLIBCXX\_HAVE\_SYS\_UIO\_H
- #define \_GLIBCXX\_HAVE\_TANF
- #define \_GLIBCXX\_HAVE\_TANHF
- #define \_GLIBCXX\_HAVE\_TANHL
- #define \_GLIBCXX\_HAVE\_TANL
- #define \_GLIBCXX\_HAVE\_TGMATH\_H
- #define \_GLIBCXX\_HAVE\_TLS
- #define \_GLIBCXX\_HAVE\_UNISTD\_H
- #define \_GLIBCXX\_HAVE\_USLEEP
- #define \_GLIBCXX\_HAVE\_VFWSCANF
- #define \_GLIBCXX\_HAVE\_VSWSCANF
- #define \_GLIBCXX\_HAVE\_VWSCANF
- #define \_GLIBCXX\_HAVE\_WCHAR\_H
- #define \_GLIBCXX\_HAVE\_WCSTOF
- #define \_GLIBCXX\_HAVE\_WCTYPE\_H
- #define \_GLIBCXX\_HAVE\_WRITEV
- #define \_GLIBCXX\_HOSTED
- #define \_GLIBCXX\_ICONV\_CONST
- #define \_GLIBCXX\_INLINE\_VERSION
- #define \_GLIBCXX\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_PACKAGE \_GLIBCXX\_VERSION
- #define \_GLIBCXX\_PACKAGE\_BUGREPORT
- #define \_GLIBCXX\_PACKAGE\_NAME
- #define \_GLIBCXX\_PACKAGE\_STRING
- #define \_GLIBCXX\_PACKAGE\_TARNAME

- `#define _GLIBCXX_PACKAGE_URL`
- `#define _GLIBCXX_PSEUDO_VISIBILITY(V)`
- `#define _GLIBCXX_RES_LIMITS`
- `#define _GLIBCXX_STDIO_EOF`
- `#define _GLIBCXX_STDIO_SEEK_CUR`
- `#define _GLIBCXX_STDIO_SEEK_END`
- `#define _GLIBCXX_SYMVER`
- `#define _GLIBCXX_SYMVER_GNU`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER(A)`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE(A)`
- `#define _GLIBCXX_THROW_OR_ABORT(_EXC)`
- `#define _GLIBCXX_USE_C99`
- `#define _GLIBCXX_USE_C99_COMPLEX`
- `#define _GLIBCXX_USE_C99_COMPLEX_TR1`
- `#define _GLIBCXX_USE_C99_CTYPE_TR1`
- `#define _GLIBCXX_USE_C99_FENV_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- `#define _GLIBCXX_USE_C99_MATH`
- `#define _GLIBCXX_USE_C99_MATH_TR1`
- `#define _GLIBCXX_USE_C99_STDINT_TR1`
- `#define _GLIBCXX_USE_DECIMAL_FLOAT`
- `#define _GLIBCXX_USE_DEPRECATED`
- `#define _GLIBCXX_USE_FLOAT128`
- `#define _GLIBCXX_USE_GET_NPROCS`
- `#define _GLIBCXX_USE_GETTIMEOFDAY`
- `#define _GLIBCXX_USE_INT128`
- `#define _GLIBCXX_USE_LFS`
- `#define _GLIBCXX_USE_LONG_LONG`
- `#define _GLIBCXX_USE_NLS`
- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_SC_NPROCESSORS_ONLN`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_VERBOSE`
- `#define _GLIBCXX_VISIBILITY(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define _GLIBCXX_X86_RDRAND`
- `#define _GTHREAD_USE_MUTEX_TIMEDLOCK`
- `#define LT_OBJDIR`
- `#define STDC_HEADERS`

#### Typedefs

- `typedef __PTRDIFF_TYPE__ std::ptrdiff_t`
- `typedef __SIZE_TYPE__ std::size_t`

#### Variables

- `decltype(nullptr) typedef std::nullptr_t`

## 5.45.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [c++config.h](#).

5.46 `c++io.h` File Reference

## Namespaces

- namespace [std](#)

## Typedefs

- typedef FILE **std::\_\_c\_file**
- typedef `__gthread_mutex_t` **std::\_\_c\_lock**

## 5.46.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [c++io.h](#).

5.47 `c++locale.h` File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define` **\_GLIBCXX\_C\_LOCALE\_GNU**
- `#define` **\_GLIBCXX\_NUM\_CATEGORIES**

## Typedefs

- typedef `__locale_t` **std::\_\_c\_locale**

## Functions

- int **std::\_\_convert\_from\_v** (const `__c_locale` &`__cloc` `__attribute__((__unused__))`, char \*`__out`, const int `__size` `__attribute__((__unused__))`, const char \*`__fmt`,...)

## 5.47.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [c++locale.h](#).

## 5.48 `c++locale_internal.h` File Reference

### 5.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [c++locale\\_internal.h](#).

## 5.49 `cassert` File Reference

### 5.49.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cassert](#).

## 5.50 `cast.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_Caster<\\_ToType>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType>  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (_FromType *__arg)`

### 5.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

Definition in file [cast.h](#).

## 5.51 cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp File Reference

### 5.51.1 Detailed Description

Contains a resize trigger implementation.

Definition in file [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger\\_imp.hpp](#).

## 5.52 cc\_ht\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CC_HASH_NAME`
- `#define PB_DS_CC_HASH_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_HASH_FN_C_DEC`

### 5.52.1 Detailed Description

Contains an implementation class for `cc_ht_map_`.

Definition in file [cc\\_ht\\_map\\_.hpp](#).

## 5.53 ccomplex File Reference

### Macros

- `#define _GLIBCXX_CCOMPLEX`

#### 5.53.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ccomplex](#).

## 5.54 `ccomplex` File Reference

### Macros

- `#define _GLIBCXX_TR1_CCOMPLEX`

#### 5.54.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ccomplex](#).

## 5.55 `cctype` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CCTYPE`

#### 5.55.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `cctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cctype](#).

## 5.56 `cctype` File Reference

### Macros

- `#define _GLIBCXX_TR1_CCTYPE`

#### 5.56.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cctype](#).

## 5.57 cerrno File Reference

### Macros

- `#define _GLIBCXX_CERRNO`
- `#define errno`

#### 5.57.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cerrno](#).

## 5.58 cfnv File Reference

### Macros

- `#define _GLIBCXX_CFENV`

#### 5.58.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cfnv](#).

## 5.59 cfnv File Reference

### Macros

- `#define _GLIBCXX_TR1_CFENV`

#### 5.59.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfnv](#).

## 5.60 cfloat File Reference

### Macros

- `#define _GLIBCXX_CFLOAT`
- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 5.60.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cfloat](#).

## 5.61 cfloat File Reference

### Macros

- `#define _GLIBCXX_TR1_CFLOAT`

### 5.61.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfloat](#).

## 5.62 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Char\\_types<\\_CharT>](#)  
*Mapping from character type to associated types.*
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)  
*Base class used to implement `std::char_traits`.*
- struct [std::char\\_traits<\\_CharT>](#)  
*Basis for explicit traits specializations.*
- struct [std::char\\_traits<char>](#)  
*21.1.3.1 char\_traits specializations*
- struct [std::char\\_traits<wchar\\_t>](#)  
*21.1.3.2 char\_traits specializations*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### 5.62.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [char\\_traits.h](#).

## 5.63 checkers.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename [\\_Iter](#) , typename [\\_Compare](#) >  
bool [\\_\\_gnu\\_parallel::\\_\\_is\\_sorted](#) ([\\_Iter](#) [\\_\\_begin](#), [\\_Iter](#) [\\_\\_end](#), [\\_Compare](#) [\\_\\_comp](#))

#### 5.63.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [checkers.h](#).

## 5.64 chrono File Reference

### Classes

- struct [std::chrono::duration](#)< [\\_Rep](#), [\\_Period](#) >  
*duration*
- struct [std::chrono::duration\\_values](#)< [\\_Rep](#) >  
*duration\_values*
- struct [std::chrono::system\\_clock](#)  
*system\_clock*
- struct [std::chrono::time\\_point](#)< [\\_Clock](#), [\\_Dur](#) >  
*time\_point*
- struct [std::chrono::treat\\_as\\_floating\\_point](#)< [\\_Rep](#) >  
*treat\_as\_floating\_point*

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

### Macros

- `#define \_GLIBCXX\_CHRONO`

### Typedefs

- typedef [system\\_clock](#) [std::chrono::high\\_resolution\\_clock](#)
- typedef [duration](#)< int, ratio< 3600 > > [std::chrono::hours](#)
- typedef [duration](#)< [int64\\_t](#), micro > [std::chrono::microseconds](#)
- typedef [duration](#)< [int64\\_t](#), milli > [std::chrono::milliseconds](#)
- typedef [duration](#)< int, ratio< 60 > > [std::chrono::minutes](#)

- typedef duration< int64\_t, nano > [std::chrono::nanoseconds](#)
- typedef duration< int64\_t > [std::chrono::seconds](#)
- typedef system\_clock **std::chrono::steady\_clock**

## Functions

- template<typename \_ToDur, typename \_Rep, typename \_Period >  
constexpr enable\_if  
< \_\_is\_duration< \_ToDur >  
::value, \_ToDur >::type [std::chrono::duration\\_cast](#) (const duration< \_Rep, \_Period > &\_\_d)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
constexpr bool **std::chrono::operator!=** (const duration< \_Rep1, \_Period1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Dur2 >  
constexpr bool **std::chrono::operator!=** (const time\_point< \_Clock, \_Dur1 > &\_\_lhs, const time\_point< \_Clock, \_Dur2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
constexpr duration< typename  
\_\_common\_rep\_type< \_Rep1,  
typename enable\_if  
<!\_\_is\_duration< \_Rep2 >  
::value, \_Rep2 >::type >::type,  
\_Period > **std::chrono::operator%** (const duration< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
constexpr [common\\_type](#)  
< duration< \_Rep1, \_Period1 >  
, duration< \_Rep2, \_Period2 >  
>::type **std::chrono::operator%** (const duration< \_Rep1, \_Period1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
constexpr duration< typename  
\_\_common\_rep\_type< \_Rep1,  
\_Rep2 >::type, \_Period > **std::chrono::operator\*** (const duration< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Rep2, typename \_Period >  
constexpr duration< typename  
\_\_common\_rep\_type< \_Rep2,  
\_Rep1 >::type, \_Period > **std::chrono::operator\*** (const \_Rep1 &\_\_s, const duration< \_Rep2, \_Period > &\_\_d)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
constexpr [common\\_type](#)  
< duration< \_Rep1, \_Period1 >  
, duration< \_Rep2, \_Period2 >  
>::type **std::chrono::operator+** (const duration< \_Rep1, \_Period1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Rep2, typename \_Period2 >  
constexpr time\_point< \_Clock,  
typename [common\\_type](#)< \_Dur1,  
duration< \_Rep2, \_Period2 >  
>::type > **std::chrono::operator+** (const time\_point< \_Clock, \_Dur1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)

- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >`  
`constexpr time_point< _Clock,`  
`typename common_type< duration`  
`< _Rep1, _Period1 >, _Dur2 >`  
`::type > std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2`  
`> &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type`  
`< duration< _Rep1, _Period1 >`  
`, duration< _Rep2, _Period2 >`  
`>::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2`  
`> &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`constexpr time_point< _Clock,`  
`typename common_type< _Dur1,`  
`duration< _Rep2, _Period2 >`  
`>::type > std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _-`  
`Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr common_type< _Dur1,`  
`_Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`constexpr duration< typename`  
`__common_rep_type< _Rep1,`  
`typename enable_if`  
`<! __is_duration< _Rep2 >`  
`::value, _Rep2 >::type >::type,`  
`_Period > std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< _Rep1,`  
`_Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _-`  
`Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator== (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`

- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator>= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _-`  
`_Clock, _Dur2 > &__rhs)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`  
`constexpr enable_if`  
`< __is_duration< _ToDur >`  
`::value, time_point< _Clock,`  
`_ToDur > >::type std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)`

#### 5.64.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [chrono](#).

## 5.65 cinttypes File Reference

### Macros

- `#define _GLIBCXX_CINTTYPES`

#### 5.65.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cinttypes](#).

## 5.66 cinttypes File Reference

### Macros

- `#define _GLIBCXX_TR1_CINTTYPES`

#### 5.66.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cinttypes](#).

## 5.67 ciso646 File Reference

#### 5.67.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ciso646](#).

## 5.68 `climits` File Reference

### Macros

- `#define _GLIBCXX_CLIMITS`
- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

#### 5.68.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [climits](#).

## 5.69 `climits` File Reference

### Macros

- `#define _GLIBCXX_TR1_CLIMITS`

#### 5.69.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/climits](#).

## 5.70 `clocale` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CLOCALE`

#### 5.70.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [clocale](#).

## 5.71 cmath File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CMATH`

### Functions

- constexpr double **std::abs** (double \_\_x)
- constexpr float **std::abs** (float \_\_x)
- constexpr long double **std::abs** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::abs** (\_Tp \_\_x)
- constexpr float **std::acos** (float \_\_x)
- constexpr long double **std::acos** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::acos** (\_Tp \_\_x)
- constexpr float **std::asin** (float \_\_x)
- constexpr long double **std::asin** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::asin** (\_Tp \_\_x)
- constexpr float **std::atan** (float \_\_x)
- constexpr long double **std::atan** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::atan** (\_Tp \_\_x)
- constexpr float **std::atan2** (float \_\_y, float \_\_x)
- constexpr long double **std::atan2** (long double \_\_y, long double \_\_x)
- template<typename \_Tp, typename \_Up >  
constexpr  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp,  
\_Up >::\_\_type **std::atan2** (\_Tp \_\_y, \_Up \_\_x)

- constexpr float **std::ceil** (float \_\_x)
- constexpr long double **std::ceil** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::ceil** (\_Tp \_\_x)
- constexpr float **std::cos** (float \_\_x)
- constexpr long double **std::cos** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::cos** (\_Tp \_\_x)
- constexpr float **std::cosh** (float \_\_x)
- constexpr long double **std::cosh** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::cosh** (\_Tp \_\_x)
- constexpr float **std::exp** (float \_\_x)
- constexpr long double **std::exp** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::exp** (\_Tp \_\_x)
- constexpr float **std::fabs** (float \_\_x)
- constexpr long double **std::fabs** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::fabs** (\_Tp \_\_x)
- constexpr float **std::floor** (float \_\_x)
- constexpr long double **std::floor** (long double \_\_x)
- template<typename \_Tp >  
constexpr  
\_\_gnu\_cxx::\_\_enable\_if  
< \_\_is\_integer< \_Tp >::\_\_value,  
double >::\_\_type **std::floor** (\_Tp \_\_x)
- constexpr float **std::fmod** (float \_\_x, float \_\_y)
- constexpr long double **std::fmod** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
constexpr  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp,  
\_Up >::\_\_type **std::fmod** (\_Tp \_\_x, \_Up \_\_y)
- float **std::frexp** (float \_\_x, int \*\_\_exp)
- long double **std::frexp** (long double \_\_x, int \*\_\_exp)

- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::frexp ( _Tp __x, int *__exp)`
- `constexpr float std::ldexp (float __x, int __exp)`
- `constexpr long double std::ldexp (long double __x, int __exp)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::ldexp ( _Tp __x, int __exp)`
- `constexpr float std::log (float __x)`
- `constexpr long double std::log (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::log ( _Tp __x)`
- `constexpr float std::log10 (float __x)`
- `constexpr long double std::log10 (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::log10 ( _Tp __x)`
- `float std::modf (float __x, float *__iptr)`
- `long double std::modf (long double __x, long double *__iptr)`
- `constexpr float std::pow (float __x, float __y)`
- `constexpr long double std::pow (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`constexpr`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type std::pow ( _Tp __x, _Up __y)`
- `constexpr float std::sin (float __x)`
- `constexpr long double std::sin (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::sin ( _Tp __x)`
- `constexpr float std::sinh (float __x)`
- `constexpr long double std::sinh (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::sinh ( _Tp __x)`
- `constexpr float std::sqrt (float __x)`
- `constexpr long double std::sqrt (long double __x)`

- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::sqrt (_Tp __x)`
- `constexpr float std::tan (float __x)`
- `constexpr long double std::tan (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::tan (_Tp __x)`
- `constexpr float std::tanh (float __x)`
- `constexpr long double std::tanh (long double __x)`
- `template<typename _Tp >`  
`constexpr`  
`__gnu_cxx::__enable_if`  
`< __is_integer< _Tp >::__value,`  
`double >::__type std::tanh (_Tp __x)`

#### 5.71.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cmath](#).

## 5.72 cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CMATH`

### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::tr1::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::tr1::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`

- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx,`  
`_Tpy >::__type std::tr1::beta ( _Tpx __x, _Tpy __y)`
- `float std::tr1::betaf (float __x, float __y)`
- `long double std::tr1::betal (long double __x, long double __y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_1 ( _Tp __k)`
- `float std::tr1::comp\_ellint\_1f (float __k)`
- `long double std::tr1::comp\_ellint\_1l (long double __k)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_2 ( _Tp __k)`
- `float std::tr1::comp\_ellint\_2f (float __k)`
- `long double std::tr1::comp\_ellint\_2l (long double __k)`
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Tpn >::__type std::tr1::comp\_ellint\_3 ( _Tp __k, _Tpn __nu)`
- `float std::tr1::comp\_ellint\_3f (float __k, float __nu)`
- `long double std::tr1::comp\_ellint\_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa,`  
`_Tpc, _Tp >::__type std::tr1::conf\_hyperg ( _Tpa __a, _Tpc __c, _Tp __x)`
- `float std::tr1::conf\_hypergf (float __a, float __c, float __x)`
- `long double std::tr1::conf\_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu,`  
`_Tp >::__type std::tr1::cyl\_bessel\_i ( _Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl\_bessel\_if (float __nu, float __x)`
- `long double std::tr1::cyl\_bessel\_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu,`  
`_Tp >::__type std::tr1::cyl\_bessel\_j ( _Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl\_bessel\_jf (float __nu, float __x)`
- `long double std::tr1::cyl\_bessel\_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu,`  
`_Tp >::__type std::tr1::cyl\_bessel\_k ( _Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl\_bessel\_kf (float __nu, float __x)`
- `long double std::tr1::cyl\_bessel\_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu,`  
`_Tp >::__type std::tr1::cyl\_neumann ( _Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl\_neumannf (float __nu, float __x)`
- `long double std::tr1::cyl\_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Tpp >::__type std::tr1::ellint\_1 ( _Tp __k, _Tpp __phi)`
- `float std::tr1::ellint\_1f (float __k, float __phi)`
- `long double std::tr1::ellint\_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Tpp >::__type std::tr1::ellint\_2 ( _Tp __k, _Tpp __phi)`
- `float std::tr1::ellint\_2f (float __k, float __phi)`

- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp,  
\_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa,  
\_Tpb, \_Tpc, \_Tp >::\_\_type **std::tr1::hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **std::tr1::hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- double **std::tr1::pow** (double \_\_x, double \_\_y)
- float **std::tr1::powf** (float \_\_x, float \_\_y)
- long double **std::tr1::powl** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp,  
\_Up >::\_\_type **std::tr1::pow** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

## 5.72.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cmath](#).

5.73 `cmp_fn_imps.hpp` File Reference

## 5.73.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container comparison related functions.

Definition in file [cmp\\_fn\\_imps.hpp](#).

5.74 `codecvt.h` File Reference

## Classes

- class [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, \\_StateT >](#)  
*Common base for codecvt functions.*
- class [std::codecvt< \\_InternT, \\_ExternT, \\_StateT >](#)  
*Primary class template codecvt.*  
*NB: Generic, mostly useless implementation.*
- class [std::codecvt< char, char, mbstate\\_t >](#)  
*class codecvt<char, char, mbstate\_t> specialization.*
- class [std::codecvt< wchar\\_t, char, mbstate\\_t >](#)  
*class codecvt<wchar\_t, char, mbstate\_t> specialization.*
- class [std::codecvt\\_base](#)  
*Empty base class for codecvt facet [22.2.1.5].*
- class [std::codecvt\\_byname< \\_InternT, \\_ExternT, \\_StateT >](#)  
*class codecvt\_byname [22.2.1.6].*

## Namespaces

- namespace [std](#)

## 5.74.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [codecvt.h](#).

5.75 `codecvt_specializations.h` File Reference

## Classes

- struct [\\_\\_gnu\\_cxx::encoding\\_char\\_traits< \\_CharT >](#)  
*encoding\_char\_traits*

- class [\\_\\_gnu\\_cxx::encoding\\_state](#)  
*Extension to use iconv for dealing with character encodings.*
- class [std::codecvt<\\_InternT, \\_ExternT, encoding\\_state>](#)  
*codecvt<InternT, \_ExternT, encoding\_state> specialization.*

#### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

#### Functions

- template<typename \_Tp >  
size\_t **std::\_\_iconv\_adaptor** (size\_t(\*\_\_func)(iconv\_t, \_Tp, size\_t \*, char \*\*, size\_t \*), iconv\_t \_\_cd, char \*\*\_\_inbuf, size\_t \*\_\_inbytes, char \*\*\_\_outbuf, size\_t \*\_\_outbytes)

##### 5.75.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [codecvt\\_specializations.h](#).

## 5.76 compatibility.h File Reference

##### 5.76.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

Definition in file [x86\\_64-unknown-linux-gnu/bits/compatibility.h](#).

## 5.77 compatibility.h File Reference

#### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

#### Functions

- template<typename \_Tp >  
\_Tp **\_\_gnu\_parallel::\_\_add\_omp** (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- template<typename \_Tp >  
bool **\_\_gnu\_parallel::\_\_cas\_omp** (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Tp >  
bool **\_\_gnu\_parallel::\_\_compare\_and\_swap** (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Tp >  
\_Tp **\_\_gnu\_parallel::\_\_fetch\_and\_add** (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- void **\_\_gnu\_parallel::\_\_yield** ()

## 5.77.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

Definition in file [parallel/compatibility.h](#).

## 5.78 compiletime\_settings.h File Reference

## Macros

- [#define \\_GLIBCXX\\_ASSERTIONS](#)
- [#define \\_GLIBCXX\\_CALL\(\\_\\_n\)](#)
- [#define \\_GLIBCXX\\_RANDOM\\_SHUFFLE\\_CONSIDER\\_L1](#)
- [#define \\_GLIBCXX\\_RANDOM\\_SHUFFLE\\_CONSIDER\\_TLB](#)
- [#define \\_GLIBCXX\\_SCALE\\_DOWN\\_FPU](#)
- [#define \\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

## 5.78.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [compiletime\\_settings.h](#).

## 5.78.2 Macro Definition Documentation

## 5.78.2.1 #define \_GLIBCXX\_ASSERTIONS

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

## 5.78.2.2 #define \_GLIBCXX\_CALL( \_\_n )

Macro to produce log message when entering a function.

## Parameters

|                  |             |
|------------------|-------------|
| <code>__n</code> | Input size. |
|------------------|-------------|

## See Also

[\\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

Definition at line 44 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_`

`parallel::multiway_merge_3_variant()`, `__gnu_parallel::multiway_merge_4_variant()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::multiway_merge_loser_tree_unguarded()`, `__gnu_parallel::multiway_merge_sentinels()`, `__gnu_parallel::parallel_multiway_merge()`, and `__gnu_parallel::parallel_sort_mwms()`.

#### 5.78.2.3 `#define GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the L1 cache for `gnu_parallel::parallel_random_shuffle()`.

Definition at line 68 of file `comptime_settings.h`.

#### 5.78.2.4 `#define GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel::parallel_random_shuffle()`.

Definition at line 74 of file `comptime_settings.h`.

#### 5.78.2.5 `#define GLIBCXX_SCALE_DOWN_FPU`

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `comptime_settings.h`.

#### 5.78.2.6 `#define GLIBCXX_VERBOSE_LEVEL`

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `comptime_settings.h`.

## 5.79 complex File Reference

### Classes

- struct [std::complex< \\_Tp >](#)
- struct [std::complex< double >](#)  
*26.2.3 complex specializations complex<double> specialization*
- struct [std::complex< float >](#)  
*26.2.3 complex specializations complex<float> specialization*
- struct [std::complex< long double >](#)  
*26.2.3 complex specializations complex<long double> specialization*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Macros

- `#define _GLIBCXX_COMPLEX`

## Functions

- `template<typename _Tp >`  
`_Tp std::__complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::__complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::asin (const std::complex< _Tp > &__z)`

- `template<typename _Tp >`  
`std::complex< _Tp > std::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::pow (const std::complex< _Tp > &__x, const _Up &__y)`

- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::proj (_Tp __x)`
- `template<typename _Tp >`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

### 5.79.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex](#).

## 5.80 complex File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define \_GLIBCXX\_TR1\_COMPLEX`

### Functions

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`

- `template<typename _Tp >`  
`std::complex< typename`  
`__gnu_cxx::__promote< _Tp >`  
`::__type > std::tr1::conj ( _Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename`  
`__gnu_cxx::__promote_2< _Tp,`  
`_Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`

#### 5.80.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/complex](#).

## 5.81 `complex.h` File Reference

### Macros

- `#define _GLIBCXX_COMPLEX_H`

#### 5.81.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex.h](#).

## 5.82 `concept_check.h` File Reference

### Macros

- `#define __glibcxx_class_requires(_a, _b)`

- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

#### 5.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [concept\\_check.h](#).

## 5.83 `concurrency.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_throw\\_concurrency\\_lock\\_error](#) ()
- void [\\_\\_gnu\\_cxx::\\_\\_throw\\_concurrency\\_unlock\\_error](#) ()

### Variables

- static const `_Lock_policy` [\\_\\_gnu\\_cxx::\\_\\_default\\_lock\\_policy](#)

#### 5.83.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [concurrency.h](#).

## 5.84 `cond_dealtor.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor](#)< `Entry`, `_Alloc` >  
*Conditional deallocate constructor argument.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.84.1 Detailed Description

Contains a conditional deallocator.

Definition in file [cond\\_dealtor.hpp](#).

## 5.85 cond\_key\_dtor\_entry\_dealtor.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#)  
*Conditional deallocate constructor argument.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.85.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

Definition in file [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#).

## 5.86 condition\_variable File Reference

## Classes

- class [std::condition\\_variable](#)  
*condition\_variable*
- class [std::condition\\_variable\\_any](#)  
*condition\_variable\_any*

## Namespaces

- namespace [std](#)

## Macros

- `#define \_GLIBCXX\_CONDITION\_VARIABLE`

## Enumerations

- enum [std::cv\\_status](#)

## 5.86.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [condition\\_variable](#).

5.87 `const_iterator.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#)  
*Const point-type iterator.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_BIN_HEAP_CIT_BASE`

## 5.87.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file [binary\\_heap\\_/const\\_iterator.hpp](#).

5.88 `const_iterator.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_< Node, \\_Alloc >](#)  
*Const point-type iterator.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_BASIC_HEAP_CIT_BASE`
- `#define PB_DS_CLASS_C_DEC`

## 5.88.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/const\\_iterator.hpp](#).

## 5.89 `const_iterator.hpp` File Reference

### Classes

- class `const_iterator_`  
*Const range-type iterator.*

#### 5.89.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.

Definition in file [unordered\\_iterator/const\\_iterator.hpp](#).

## 5.90 `constructor_destructor_fn_imps.hpp` File Reference

#### 5.90.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/constructor\\_destructor\\_fn\\_imps.hpp](#).

## 5.91 `constructor_destructor_fn_imps.hpp` File Reference

#### 5.91.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/constructor\\_destructor\\_fn\\_imps.hpp](#).

## 5.92 `constructor_destructor_fn_imps.hpp` File Reference

## 5.93 `constructor_destructor_no_store_hash_fn_imps.hpp` File Reference

#### 5.93.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/constructor\\_destructor\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.94 `constructor_destructor_no_store_hash_fn_imps.hpp` File Reference

#### 5.94.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/constructor\\_destructor\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.95 `constructor_destructor_store_hash_fn_imps.hpp` File Reference

#### 5.95.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/constructor\\_destructor\\_store\\_hash\\_fn\\_imps.hpp](#).

### 5.96 constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

#### 5.96.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/constructor\\_destructor\\_store\\_hash\\_fn\\_imps.hpp](#).

### 5.97 constructors\_destructor\_fn\_imps.hpp File Reference

#### 5.97.1 Detailed Description

Contains an implementation class for `binary_heap_`.

Definition in file [binary\\_heap\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

### 5.98 constructors\_destructor\_fn\_imps.hpp File Reference

#### 5.98.1 Detailed Description

Contains an implementation for `binomial_heap_`.

Definition in file [binomial\\_heap\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

### 5.99 constructors\_destructor\_fn\_imps.hpp File Reference

#### 5.99.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial\\_heap\\_base\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

### 5.100 constructors\_destructor\_fn\_imps.hpp File Reference

#### 5.100.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin\\_search\\_tree\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

### 5.101 constructors\_destructor\_fn\_imps.hpp File Reference

#### 5.101.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.102 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.102.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov\\_tree\\_map\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.103 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.103.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing\\_heap\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.104 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.104.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [pat\\_trie\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.105 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.105.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb\\_tree\\_map\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.106 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.106.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

Definition in file [rc\\_binomial\\_heap\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.107 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.107.1 Detailed Description

Contains an implementation class for `splay_tree_`.

Definition in file [splay\\_tree\\_/constructors\\_destructor\\_fn\\_imps.hpp](#).

## 5.108 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.108.1 Detailed Description

Contains an implementation for `thin_heap_`.

Definition in file `thin_heap_/constructors_destructor_fn_imps.hpp`.

## 5.109 container\_base\_dispatch.hpp File Reference

### Classes

- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >`  
*Specialization colision-chaining hash map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >`  
*Specialization general-probe hash map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >`  
*Specialization for list-update map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >`  
*Specialization ordered-vector tree map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >`  
*Specialization for PATRICIA trie map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >`  
*Specialization for R-B tree map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >`  
*Specialization splay tree map.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >`  
*Specialization colision-chaining hash set.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >`  
*Specialization general-probe hash set.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >`  
*Specialization for list-update set.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >`  
*Specialization ordered-vector tree set.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >`  
*Specialization for PATRICIA trie set.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI >`  
*Specialization for R-B tree set.*
- struct `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >`  
*Specialization splay tree set.*

### Namespaces

- namespace `__gnu_pbds`

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)`
- `#define PB_DS_CHECK_KEY_EXISTS(_Key)`
- `#define PB_DS_DATA_FALSE_INDICATOR`
- `#define PB_DS_DATA_TRUE_INDICATOR`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2S(X)`
- `#define PB_DS_V2S(X)`

### 5.109.1 Detailed Description

Contains associative container dispatching.

Definition in file [container\\_base\\_dispatch.hpp](#).

## 5.110 `cpp_type_traits.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### 5.110.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits>`.

Definition in file [cpp\\_type\\_traits.h](#).

## 5.111 `cpu_defines.h` File Reference

### 5.111.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [cpu\\_defines.h](#).

## 5.112 `csetjmp` File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_CSETJMP`
- `#define setjmp(env)`

### 5.112.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csetjmp](#).

## 5.113 csignal File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_CSIGNAL`

### 5.113.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csignal](#).

## 5.114 cstdarg File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_CSTDARG`
- `#define va_end(ap)`

### 5.114.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdarg](#).

## 5.115 cstdarg File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDARG`

### 5.115.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdarg](#).

## 5.116 cstdlib File Reference

### Macros

- `#define _GLIBCXX_CSTDBOOL`

### 5.116.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdlib](#).

## 5.117 cstdlib File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDBOOL`

### 5.117.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/stdlib](#).

## 5.118 cstdint File Reference

### 5.118.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdint.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdint](#).

## 5.119 cstdint File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTDINT`

#### 5.119.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdint](#).

## 5.120 cstdint File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CSTDINT`

#### 5.120.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

## 5.121 cstdio File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTDIO`

### Functions

- `char * gets (char * __s) __attribute__((deprecated))`

### 5.121.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdio](#).

## 5.122 `cstdio` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDIO`

### 5.122.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdio](#).

## 5.123 `cstdlib` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTDLIB`
- `#define EXIT_FAILURE`
- `#define EXIT_SUCCESS`

### Functions

- void **`std::abort`** (void) throw ()
- int **`std::atexit`** (void(\*)()) throw ()
- void **`std::exit`** (int) throw ()

### 5.123.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdlib](#).

## 5.124 cstdlib File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDLIB`

#### 5.124.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdlib](#).

## 5.125 cstring File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTRING`

### Functions

- `void * std::memchr (void *__s, int __c, size_t __n)`
- `char * std::strchr (char *__s, int __n)`
- `char * std::strpbrk (char *__s1, const char *__s2)`
- `char * std::strrchr (char *__s, int __n)`
- `char * std::strstr (char *__s1, const char *__s2)`

#### 5.125.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstring](#).

## 5.126 ctgmath File Reference

### Macros

- `#define _GLIBCXX_CTGMATH`

#### 5.126.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ctgmath](#).

## 5.127 ctgmath File Reference

### Macros

- `#define _GLIBCXX_TR1_CTGMATH`

#### 5.127.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctgmath](#).

## 5.128 ctime File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CTIME`

#### 5.128.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ctime](#).

## 5.129 ctime File Reference

### Macros

- `#define _GLIBCXX_TR1_CTIME`

#### 5.129.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctime](#).

## 5.130 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)  
*Base class for ctype.*

## Namespaces

- namespace [std](#)

## 5.130.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [ctype\\_base.h](#).

5.131 `ctype_inline.h` File Reference

## Namespaces

- namespace [std](#)

## 5.131.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [ctype\\_inline.h](#).

5.132 `wchar` File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_CWCHAR`

## Functions

- `wchar_t* std::wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t* std::wcsbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t* std::wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t* std::wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t* std::wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`

## 5.132.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [wchar](#).

## 5.133 `cwchar` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CWCHAR`

#### 5.133.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwchar](#).

## 5.134 `cwctype` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CWCTYPE`

#### 5.134.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwctype](#).

## 5.135 `cwctype` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CWCTYPE`

## 5.135.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwctype](#).

## 5.136 cxxabi.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)  
*Exception thrown by `__cxa_guard_acquire`.  
6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [abi](#)

## Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type)(void *)`

## Functions

- `__cxa_dependent_exception * __cxxabiv1::__cxa_allocate_dependent_exception ()` noexcept
- `void * __cxxabiv1::__cxa_allocate_exception (size_t)` noexcept
- `int __cxxabiv1::__cxa_atexit (void (*)(void *), void *, void *)` noexcept
- `void __cxxabiv1::__cxa_bad_cast ()` \_\_attribute\_\_((\_\_noreturn\_\_))
- `void __cxxabiv1::__cxa_bad_typeid ()` \_\_attribute\_\_((\_\_noreturn\_\_))
- `void * __cxxabiv1::__cxa_begin_catch (void *)` noexcept
- `std::type_info * __cxxabiv1::__cxa_current_exception_type ()` noexcept \_\_attribute\_\_((\_\_pure\_\_))
- `void __cxxabiv1::__cxa_deleted_virtual (void)` \_\_attribute\_\_((\_\_noreturn\_\_))
- `char * __cxxabiv1::__cxa_demangle (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)`
- `void __cxxabiv1::__cxa_end_catch ()`
- `int __cxxabiv1::__cxa_finalize (void *)`
- `void __cxxabiv1::__cxa_free_dependent_exception (__cxa_dependent_exception *)` noexcept
- `void __cxxabiv1::__cxa_free_exception (void *)` noexcept
- `void * __cxxabiv1::__cxa_get_exception_ptr (void *)` noexcept \_\_attribute\_\_((\_\_pure\_\_))
- `__cxa_eh_globals * __cxxabiv1::__cxa_get_globals ()` noexcept \_\_attribute\_\_((\_\_const\_\_))
- `__cxa_eh_globals * __cxxabiv1::__cxa_get_globals_fast ()` noexcept \_\_attribute\_\_((\_\_const\_\_))
- `void __cxxabiv1::__cxa_guard_abort (__guard *)` noexcept
- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *)` noexcept
- `void __cxxabiv1::__cxa_pure_virtual (void)` \_\_attribute\_\_((\_\_noreturn\_\_))
- `void __cxxabiv1::__cxa_rethrow ()` \_\_attribute\_\_((\_\_noreturn\_\_))
- `int __cxxabiv1::__cxa_thread_atexit (void (*)(void *), void *, void *)` noexcept
- `void __cxxabiv1::__cxa_throw (void *, std::type_info *, void (*)(void *))` \_\_attribute\_\_((\_\_noreturn\_\_))

- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor` (void \*\_\_dest\_array, void \*\_\_src\_array, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_ctor\_return\_type(\*\_\_constructor)(void \*, void \*), \_\_cxa\_ctor\_type \_\_destructor)
- `void __cxxabiv1::__cxa_vec_cleanup` (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_s, \_\_cxa\_ctor\_type \_\_destructor) noexcept
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor` (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor)
- `void __cxxabiv1::__cxa_vec_delete` (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type \_\_destructor)
- `void __cxxabiv1::__cxa_vec_delete2` (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type \_\_destructor, void(\*\_\_dealloc)(void \*))
- `void __cxxabiv1::__cxa_vec_delete3` (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type \_\_destructor, void(\*\_\_dealloc)(void \*, size\_t))
- `void __cxxabiv1::__cxa_vec_dtor` (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_ctor\_type \_\_destructor)
- `void * __cxxabiv1::__cxa_vec_new` (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor)
- `void * __cxxabiv1::__cxa_vec_new2` (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor, void \*(\*\_\_alloc)(size\_t), void(\*\_\_dealloc)(void \*))
- `void * __cxxabiv1::__cxa_vec_new3` (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor, void \*(\*\_\_alloc)(size\_t), void(\*\_\_dealloc)(void \*, size\_t))
- `void * __cxxabiv1::__dynamic_cast` (const void \*\_\_src\_ptr, const \_\_class\_type\_info \*\_\_src\_type, const \_\_class\_type\_info \*\_\_dst\_type, ptrdiff\_t \_\_src2dst)

#### 5.136.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi.h](#).

## 5.137 cxxabi\_forced.h File Reference

### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)  
*Thrown as part of forced unwinding.  
A magic placeholder class that can be caught by reference to recognize forced unwinding.*

#### 5.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

Definition in file [cxxabi\\_forced.h](#).

## 5.138 cxxabi\_tweaks.h File Reference

### Macros

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`

- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

#### Typedefs

- `typedef void __cxxabiv1::__cxa_ctor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

#### Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI__)))`

##### 5.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

Definition in file [cxxabi\\_tweaks.h](#).

## 5.139 debug.h File Reference

#### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

#### Macros

- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_non_empty_range(_First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_subscript(_N)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`

- `#define _GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define _GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define _GLIBCXX_DEBUG_PEDASSERT(_Condition)`

#### 5.139.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug.h](#).

## 5.140 `debug_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator<\\_Alloc>](#)  
*A meta-allocator with debugging bits, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 5.140.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [debug\\_allocator.h](#).

## 5.141 `debug_fn_imps.hpp` File Reference

#### 5.141.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary\\_heap\\_/debug\\_fn\\_imps.hpp](#).

## 5.142 `debug_fn_imps.hpp` File Reference

#### 5.142.1 Detailed Description

Contains an implementation for `binomial_heap_`.

Definition in file [binomial\\_heap\\_/debug\\_fn\\_imps.hpp](#).

## 5.143 `debug_fn_imps.hpp` File Reference

#### 5.143.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial\\_heap\\_base\\_/debug\\_fn\\_imps.hpp](#).

## 5.144 `debug_fn_imps.hpp` File Reference

### 5.144.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin\\_search\\_tree\\_/debug\\_fn\\_imps.hpp](#).

## 5.145 `debug_fn_imps.hpp` File Reference

### 5.145.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

Definition in file [cc\\_hash\\_table\\_map\\_/debug\\_fn\\_imps.hpp](#).

## 5.146 `debug_fn_imps.hpp` File Reference

### 5.146.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

Definition in file [gp\\_hash\\_table\\_map\\_/debug\\_fn\\_imps.hpp](#).

## 5.147 `debug_fn_imps.hpp` File Reference

### 5.147.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/debug\\_fn\\_imps.hpp](#).

## 5.148 `debug_fn_imps.hpp` File Reference

### 5.148.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

Definition in file [list\\_update\\_map\\_/debug\\_fn\\_imps.hpp](#).

## 5.149 `debug_fn_imps.hpp` File Reference

### 5.149.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov\\_tree\\_map\\_/debug\\_fn\\_imps.hpp](#).

## 5.150 `debug_fn_imps.hpp` File Reference

#### 5.150.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing\\_heap\\_/debug\\_fn\\_imps.hpp](#).

### 5.151 `debug_fn_imps.hpp` File Reference

#### 5.151.1 Detailed Description

Contains an implementation class for `pat_trie_`.

Definition in file [pat\\_trie\\_/debug\\_fn\\_imps.hpp](#).

### 5.152 `debug_fn_imps.hpp` File Reference

#### 5.152.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb\\_tree\\_map\\_/debug\\_fn\\_imps.hpp](#).

### 5.153 `debug_fn_imps.hpp` File Reference

#### 5.153.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

Definition in file [rc\\_binomial\\_heap\\_/debug\\_fn\\_imps.hpp](#).

### 5.154 `debug_fn_imps.hpp` File Reference

#### 5.154.1 Detailed Description

Contains an implementation class for `splay_tree_`.

Definition in file [splay\\_tree\\_/debug\\_fn\\_imps.hpp](#).

### 5.155 `debug_fn_imps.hpp` File Reference

#### 5.155.1 Detailed Description

Contains an implementation for `thin_heap_`.

Definition in file [thin\\_heap\\_/debug\\_fn\\_imps.hpp](#).

### 5.156 `debug_map_base.hpp` File Reference

#### 5.156.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file [debug\\_map\\_base.hpp](#).

## 5.157 `debug_no_store_hash_fn_imps.hpp` File Reference

### 5.157.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

Definition in file [cc\\_hash\\_table\\_map\\_/debug\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.158 `debug_no_store_hash_fn_imps.hpp` File Reference

### 5.158.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

Definition in file [gp\\_hash\\_table\\_map\\_/debug\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.159 `debug_store_hash_fn_imps.hpp` File Reference

### 5.159.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

Definition in file [cc\\_hash\\_table\\_map\\_/debug\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.160 `debug_store_hash_fn_imps.hpp` File Reference

### 5.160.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

Definition in file [gp\\_hash\\_table\\_map\\_/debug\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.161 `decimal` File Reference

### Classes

- class [std::decimal::decimal128](#)  
*3.2.4 Class decimal128.*
- class [std::decimal::decimal32](#)  
*3.2.2 Class decimal32.*
- class [std::decimal::decimal64](#)  
*3.2.3 Class decimal64.*

### Namespaces

- namespace [std](#)
- namespace [std::decimal](#)

## Macros

- `#define _DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_DEC(_Op, _T1, _T2, _T3)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_INT(_Op, _Tp)`
- `#define _DECLARE_DECIMAL_COMPARISON(_Op, _Tp)`
- `#define _GLIBCXX_DECIMAL`
- `#define _GLIBCXX_USE_DECIMAL_`

## Functions

- `double std::decimal::decimal128_to_double (decimal128 __d)`
- `float std::decimal::decimal128_to_float (decimal128 __d)`
- `long double std::decimal::decimal128_to_long_double (decimal128 __d)`
- `long long std::decimal::decimal128_to_long_long (decimal128 __d)`
- `double std::decimal::decimal32_to_double (decimal32 __d)`
- `float std::decimal::decimal32_to_float (decimal32 __d)`
- `long double std::decimal::decimal32_to_long_double (decimal32 __d)`
- `long long std::decimal::decimal32_to_long_long (decimal32 __d)`
- `double std::decimal::decimal64_to_double (decimal64 __d)`
- `float std::decimal::decimal64_to_float (decimal64 __d)`
- `long double std::decimal::decimal64_to_long_double (decimal64 __d)`
- `long long std::decimal::decimal64_to_long_long (decimal64 __d)`
- `double std::decimal::decimal_to_double (decimal32 __d)`
- `double std::decimal::decimal_to_double (decimal64 __d)`
- `double std::decimal::decimal_to_double (decimal128 __d)`
- `float std::decimal::decimal_to_float (decimal32 __d)`
- `float std::decimal::decimal_to_float (decimal64 __d)`
- `float std::decimal::decimal_to_float (decimal128 __d)`
- `long double std::decimal::decimal_to_long_double (decimal32 __d)`
- `long double std::decimal::decimal_to_long_double (decimal64 __d)`
- `long double std::decimal::decimal_to_long_double (decimal128 __d)`
- `long long std::decimal::decimal_to_long_long (decimal32 __d)`
- `long long std::decimal::decimal_to_long_long (decimal64 __d)`
- `long long std::decimal::decimal_to_long_long (decimal128 __d)`
- `static decimal128 std::decimal::make_decimal128 (long long __coeff, int __exp)`
- `static decimal128 std::decimal::make_decimal128 (unsigned long long __coeff, int __exp)`
- `static decimal32 std::decimal::make_decimal32 (long long __coeff, int __exp)`
- `static decimal32 std::decimal::make_decimal32 (unsigned long long __coeff, int __exp)`
- `static decimal64 std::decimal::make_decimal64 (long long __coeff, int __exp)`
- `static decimal64 std::decimal::make_decimal64 (unsigned long long __coeff, int __exp)`
- `bool std::decimal::operator!= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, unsigned long __rhs)`

- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)

- decimal64 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)

- decimal64 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator+** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator+** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator-** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)

- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)

- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)

- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal32 \_\_rhs)

- bool **std::decimal::operator==** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator>** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator>** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator>** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal128 \_\_rhs)

- `bool std::decimal::operator> (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal128 __rhs)`

## 5.161.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [decimal](#).

## 5.162 deque File Reference

## Macros

- `#define _GLIBCXX_DEQUE`

## 5.162.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [deque](#).

## 5.163 deque File Reference

## Classes

- class [std::\\_\\_debug::deque<\\_Tp, \\_Allocator >](#)  
*Class std::deque with safety/checking/debug instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_DEQUE`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

## 5.163.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/deque](#).

## 5.164 deque File Reference

## Classes

- class [std::\\_\\_profile::deque< \\_Tp, \\_Allocator >](#)  
*Class std::deque wrapper with performance instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Macros

- `#define _GLIBCXX_PROFILE_DEQUE`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

## 5.164.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/deque](#).

## 5.165 deque.tcc File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _DEQUE_TCC`

## Functions

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > std::copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > std::copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<`  
`_Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp *`  
`> &__last, const _Tp &__value)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > std::move (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > std::move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<`  
`_Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

## 5.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

Definition in file [deque.tcc](#).

## 5.166 direct\_mask\_range\_hashing\_imp.hpp File Reference

## 5.166.1 Detailed Description

Contains a range-hashing policy implementation

Definition in file [direct\\_mask\\_range\\_hashing\\_imp.hpp](#).

## 5.167 direct\_mod\_range\_hashing\_imp.hpp File Reference

## 5.167.1 Detailed Description

Contains a range-hashing policy implementation

Definition in file [direct\\_mod\\_range\\_hashing\\_imp.hpp](#).

## 5.168 dynamic\_bitset File Reference

### Classes

- struct [std::tr2::\\_\\_dynamic\\_bitset\\_base< \\_WordT, \\_Alloc >](#)
- class [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >](#)  
*The dynamic\_bitset class represents a sequence of bits.*
- class [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >::reference](#)

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)
- namespace [std::tr2::\\_\\_detail](#)

### Macros

- `#define \_GLIBCXX\_TR2\_DYNAMIC\_BITSET`

### Functions

- `template<typename _WordT, typename _Alloc >  
bool std::tr2::operator== (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >  
bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >  
bool std::tr2::operator< (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >  
bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >  
bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >  
bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >  
dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >  
dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >  
dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc >` [std::tr2::operator-](#) (`const dynamic_bitset< _WordT, _Alloc > &__x`, `const dynamic_bitset< _WordT, _Alloc > &__y`)
- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`  
[std::basic\\_istream](#)< `_CharT`,  
`_Traits` > & [std::tr2::operator>>](#) ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &\_\_is, `dynamic_bitset< _WordT, _Alloc > &__x`)
- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`  
[std::basic\\_ostream](#)< `_CharT`,  
`_Traits` > & [std::tr2::operator<<](#) ([std::basic\\_ostream](#)< `_CharT`, `_Traits` > &\_\_os, `const dynamic_bitset< _WordT, _Alloc > &__x`)

#### 5.168.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file [dynamic\\_bitset](#).

## 5.169 `enc_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf](#)< `_CharT` >  
*class `enc_filebuf`.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 5.169.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [enc\\_filebuf.h](#).

## 5.170 `entry_cmp.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp](#)< `_VTp`, `Cmp_Fn`, `_Alloc`, `false` >  
*Specialization, false.*
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp](#)< `_VTp`, `Cmp_Fn`, `_Alloc`, `false` >::type  
*Compare plus entry.*
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp](#)< `_VTp`, `Cmp_Fn`, `_Alloc`, `true` >  
*Specialization, true.*
- struct [entry\\_cmp](#)< `_VTp`, `Cmp_Fn`, `_Alloc`, `No_Throw` >  
*Entry compare, primary template.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.170.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [entry\\_cmp.hpp](#).

5.171 `entry_list_fn_imps.hpp` File Reference

## 5.171.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entry-list related functions.

Definition in file [entry\\_list\\_fn\\_imps.hpp](#).

5.172 `entry_metadata_base.hpp` File Reference

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.172.1 Detailed Description

Contains an implementation for a list update map.

Definition in file [entry\\_metadata\\_base.hpp](#).

5.173 `entry_pred.hpp` File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, false>](#)  
*Specialization, false.*
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, true>](#)  
*Specialization, true.*
- struct [entry\\_pred<\\_VTp, Pred, \\_Alloc, No\\_Throw>](#)  
*Entry predicate primary class template.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.173.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [entry\\_pred.hpp](#).

## 5.174 `eq_by_less.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::eq\\_by\\_less< Key, Cmp\\_Fn >](#)  
*Equivalence function.*

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 5.174.1 Detailed Description

Contains an equivalence function.

Definition in file [eq\\_by\\_less.hpp](#).

## 5.175 `equally_split.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename `_DifferenceType` , typename `_OutputIterator` >  
`_OutputIterator` [\\_\\_gnu\\_parallel::\\_\\_equally\\_split](#) (`_DifferenceType` `__n`, `_ThreadIndex` `__num_threads`, `_OutputIterator` `__s`)
- template<typename `_DifferenceType` >  
`_DifferenceType` [\\_\\_gnu\\_parallel::\\_\\_equally\\_split\\_point](#) (`_DifferenceType` `__n`, `_ThreadIndex` `__num_threads`, `_ThreadIndex` `__thread_no`)

#### 5.175.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [equally\\_split.h](#).

## 5.176 `erase_fn_imps.hpp` File Reference

#### 5.176.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary\\_heap\\_/erase\\_fn\\_imps.hpp](#).

## 5.177 `erase_fn_imps.hpp` File Reference

### 5.177.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial\\_heap\\_base\\_/erase\\_fn\\_imps.hpp](#).

## 5.178 `erase_fn_imps.hpp` File Reference

### 5.178.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin\\_search\\_tree\\_/erase\\_fn\\_imps.hpp](#).

## 5.179 `erase_fn_imps.hpp` File Reference

### 5.179.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/erase\\_fn\\_imps.hpp](#).

## 5.180 `erase_fn_imps.hpp` File Reference

### 5.180.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/erase\\_fn\\_imps.hpp](#).

## 5.181 `erase_fn_imps.hpp` File Reference

### 5.181.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/erase\\_fn\\_imps.hpp](#).

## 5.182 `erase_fn_imps.hpp` File Reference

### 5.182.1 Detailed Description

Contains implementations of `lu_map_`.

Definition in file [list\\_update\\_map\\_/erase\\_fn\\_imps.hpp](#).

## 5.183 `erase_fn_imps.hpp` File Reference

#### 5.183.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov\\_tree\\_map\\_/erase\\_fn\\_imps.hpp](#).

### 5.184 `erase_fn_imps.hpp` File Reference

#### 5.184.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing\\_heap\\_/erase\\_fn\\_imps.hpp](#).

### 5.185 `erase_fn_imps.hpp` File Reference

#### 5.185.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [pat\\_trie\\_/erase\\_fn\\_imps.hpp](#).

### 5.186 `erase_fn_imps.hpp` File Reference

#### 5.186.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb\\_tree\\_map\\_/erase\\_fn\\_imps.hpp](#).

### 5.187 `erase_fn_imps.hpp` File Reference

#### 5.187.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

Definition in file [rc\\_binomial\\_heap\\_/erase\\_fn\\_imps.hpp](#).

### 5.188 `erase_fn_imps.hpp` File Reference

#### 5.188.1 Detailed Description

Contains an implementation class for `splay_tree_`.

Definition in file [splay\\_tree\\_/erase\\_fn\\_imps.hpp](#).

### 5.189 `erase_fn_imps.hpp` File Reference

#### 5.189.1 Detailed Description

Contains an implementation for `thin_heap_`.

Definition in file [thin\\_heap\\_/erase\\_fn\\_imps.hpp](#).

## 5.190 `erase_no_store_hash_fn_imps.hpp` File Reference

### 5.190.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is not stored.

Definition in file [cc\\_hash\\_table\\_map\\_/erase\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.191 `erase_no_store_hash_fn_imps.hpp` File Reference

### 5.191.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is not stored.

Definition in file [gp\\_hash\\_table\\_map\\_/erase\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.192 `erase_store_hash_fn_imps.hpp` File Reference

### 5.192.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is stored.

Definition in file [cc\\_hash\\_table\\_map\\_/erase\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.193 `erase_store_hash_fn_imps.hpp` File Reference

### 5.193.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is stored.

Definition in file [gp\\_hash\\_table\\_map\\_/erase\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.194 `error_constants.h` File Reference

### Namespaces

- namespace [std](#)

### Enumerations

- enum `errc`

### 5.194.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

Definition in file [error\\_constants.h](#).

## 5.195 exception File Reference

### Classes

- class [std::bad\\_exception](#)
- class [std::exception](#)

*Base class for all library exceptions.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Typedefs

- typedef void(\* [std::terminate\\_handler](#) )()
- typedef void(\* [std::unexpected\\_handler](#) )()

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- terminate\_handler [std::set\\_terminate](#) (terminate\_handler) noexcept
- unexpected\_handler [std::set\\_unexpected](#) (unexpected\_handler) noexcept
- void [std::terminate](#) () noexcept \_\_attribute\_\_((\_\_noreturn\_\_))
- bool [std::uncaught\\_exception](#) () noexcept \_\_attribute\_\_((\_\_pure\_\_))
- void [std::unexpected](#) () \_\_attribute\_\_((\_\_noreturn\_\_))

#### 5.195.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [exception](#).

## 5.196 exception.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)

*Base class for exceptions.*

- struct [\\_\\_gnu\\_pbds::insert\\_error](#)

*An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the allocator\_type's exception will be thrown).*

- struct [\\_\\_gnu\\_pbds::join\\_error](#)

*A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps).*

- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

*A container cannot be resized.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error](#) ()
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error](#) ()
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error](#) ()
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error](#) ()

## 5.196.1 Detailed Description

Contains exception classes.

Definition in file [exception.hpp](#).

5.197 `exception_defines.h` File Reference

## Macros

- `#define` [\\_\\_catch\(X\)](#)
- `#define` [\\_\\_throw\\_exception\\_again](#)
- `#define` [\\_\\_try](#)

## 5.197.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

Definition in file [exception\\_defines.h](#).

5.198 `exception_ptr.h` File Reference

## Classes

- class [std::\\_\\_exception\\_ptr::exception\\_ptr](#)  
*An opaque pointer to an arbitrary exception.*

## Namespaces

- namespace [std](#)

## Functions

- template<typename [\\_Ex](#) >  
  [exception\\_ptr std::copy\\_exception](#) ([\\_Ex \\_\\_ex](#)) noexcept
- [exception\\_ptr std::current\\_exception](#) () noexcept

- `template<typename _Ex >`  
`exception_ptr std::make\_exception\_ptr (_Ex __ex) noexcept`
- `bool std::__exception_ptr::operator!= (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `bool std::__exception_ptr::operator== (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `void std::rethrow\_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::__exception_ptr::swap (exception_ptr &__lhs, exception_ptr &__rhs)`

#### 5.198.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

Definition in file [exception\\_ptr.h](#).

## 5.199 extc++.h File Reference

#### 5.199.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file [extc++.h](#).

## 5.200 extptr\_allocator.h File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_ExtPtr\\_allocator<\\_Tp>](#)  
*An example allocator which uses a non-standard pointer type.  
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.*

#### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### Functions

- `template<typename _Tp >`  
`void __gnu_cxx::swap (_ExtPtr_allocator<_Tp> &__larg, _ExtPtr_allocator<_Tp> &__rarg)`

#### 5.200.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

**Author**

Bob Walters

An example allocator which uses an alternative pointer type from `bits/pointer.h`. Supports test cases which confirm container support for alternative pointers.

Definition in file [extptr\\_allocator.h](#).

**5.201 features.h File Reference****Macros**

- [#define \\_GLIBCXX\\_BAL\\_QUICKSORT](#)
- [#define \\_GLIBCXX\\_FIND\\_CONSTANT\\_SIZE\\_BLOCKS](#)
- [#define \\_GLIBCXX\\_FIND\\_EQUAL\\_SPLIT](#)
- [#define \\_GLIBCXX\\_FIND\\_GROWING\\_BLOCKS](#)
- [#define \\_GLIBCXX\\_MERGESORT](#)
- [#define \\_GLIBCXX\\_QUICKSORT](#)
- [#define \\_GLIBCXX\\_TREE\\_DYNAMIC\\_BALANCING](#)
- [#define \\_GLIBCXX\\_TREE\\_FULL\\_COPY](#)
- [#define \\_GLIBCXX\\_TREE\\_INITIAL\\_SPLITTING](#)

**5.201.1 Detailed Description**

Defines on whether to include algorithm variants. Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [features.h](#).

**5.201.2 Macro Definition Documentation****5.201.2.1 #define \_GLIBCXX\_BAL\_QUICKSORT**

Include parallel dynamically load-balanced quicksort.

**See Also**

[\\_\\_gnu\\_parallel::\\_Settings::sort\\_algorithm](#)

Definition at line 55 of file `features.h`.

**5.201.2.2 #define \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS**

Include the equal-sized blocks variant for `std::find`.

**See Also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_algorithm](#)

Definition at line 67 of file `features.h`.

#### 5.201.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

##### See Also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file `features.h`.

#### 5.201.2.4 `#define _GLIBCXX_FIND_GROWING_BLOCKS`

Include the growing blocks variant for `std::find`.

##### See Also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file `features.h`.

#### 5.201.2.5 `#define _GLIBCXX_MERGESORT`

Include parallel multi-way mergesort.

##### See Also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

#### 5.201.2.6 `#define _GLIBCXX_QUICKSORT`

Include parallel unbalanced quicksort.

##### See Also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file `features.h`.

#### 5.201.2.7 `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

##### See Also

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file `features.h`.

#### 5.201.2.8 `#define _GLIBCXX_TREE_FULL_COPY`

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

##### See Also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file `features.h`.

5.201.2.9 `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See Also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

5.202 `fcntl.h` File Reference

## 5.202.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fcntl.h](#).

5.203 `find.h` File Reference

Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< `_RAIter1`, `_RAIter2` > [\\_\\_gnu\\_parallel::find\\_template](#) (`_RAIter1` \_\_begin1, `_RAIter1` \_\_end1, `_RAIter2` \_\_begin2, `_Pred` \_\_pred, `_Selector` \_\_selector)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< `_RAIter1`, `_RAIter2` > [\\_\\_gnu\\_parallel::find\\_template](#) (`_RAIter1` \_\_begin1, `_RAIter1` \_\_end1, `_RAIter2` \_\_begin2, `_Pred` \_\_pred, `_Selector` \_\_selector, `equal_split_tag`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< `_RAIter1`, `_RAIter2` > [\\_\\_gnu\\_parallel::find\\_template](#) (`_RAIter1` \_\_begin1, `_RAIter1` \_\_end1, `_RAIter2` \_\_begin2, `_Pred` \_\_pred, `_Selector` \_\_selector, `growing_blocks_tag`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< `_RAIter1`, `_RAIter2` > [\\_\\_gnu\\_parallel::find\\_template](#) (`_RAIter1` \_\_begin1, `_RAIter1` \_\_end1, `_RAIter2` \_\_begin2, `_Pred` \_\_pred, `_Selector` \_\_selector, `constant_size_blocks_tag`)

## 5.203.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find.h](#).

5.204 `find_fn_imps.hpp` File Reference

## 5.204.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary\\_heap/find\\_fn\\_imps.hpp](#).

## 5.205 find\_fn\_imps.hpp File Reference

### 5.205.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial\\_heap\\_base\\_/find\\_fn\\_imps.hpp](#).

## 5.206 find\_fn\_imps.hpp File Reference

### 5.206.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

Definition in file [bin\\_search\\_tree\\_/find\\_fn\\_imps.hpp](#).

## 5.207 find\_fn\_imps.hpp File Reference

### 5.207.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s find related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/find\\_fn\\_imps.hpp](#).

## 5.208 find\_fn\_imps.hpp File Reference

### 5.208.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s find related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/find\\_fn\\_imps.hpp](#).

## 5.209 find\_fn\_imps.hpp File Reference

### 5.209.1 Detailed Description

Contains implementations of lu\_map\_.

Definition in file [list\\_update\\_map\\_/find\\_fn\\_imps.hpp](#).

## 5.210 find\_fn\_imps.hpp File Reference

### 5.210.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing\\_heap\\_/find\\_fn\\_imps.hpp](#).

## 5.211 find\_fn\_imps.hpp File Reference

### 5.211.1 Detailed Description

Contains an implementation class for pat\_trie.

Definition in file [pat\\_trie\\_/find\\_fn\\_imps.hpp](#).

## 5.212 find\_fn\_imps.hpp File Reference

### 5.212.1 Detailed Description

Contains an implementation for rb\_tree\_.

Definition in file [rb\\_tree\\_map\\_/find\\_fn\\_imps.hpp](#).

## 5.213 find\_fn\_imps.hpp File Reference

### 5.213.1 Detailed Description

Contains an implementation class for splay\_tree\_.

Definition in file [splay\\_tree\\_/find\\_fn\\_imps.hpp](#).

## 5.214 find\_fn\_imps.hpp File Reference

### 5.214.1 Detailed Description

Contains an implementation for thin\_heap\_.

Definition in file [thin\\_heap\\_/find\\_fn\\_imps.hpp](#).

## 5.215 find\_no\_store\_hash\_fn\_imps.hpp File Reference

### 5.215.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s find related functions, when the hash value is not stored.

Definition in file [find\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.216 find\_selectors.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)  
*Test predicate on two adjacent elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>](#)  
*Test predicate on several elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)  
*Test predicate on a single element, used for std::find() and std::find\_if ().*
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)  
*Base class of all \_\_gnu\_parallel::\_\_find\_template selectors.*

- struct [\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)  
*Test inverted predicate on a single element.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 5.216.1 Detailed Description

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find\\_selectors.h](#).

## 5.217 find\_store\_hash\_fn\_imps.hpp File Reference

### 5.217.1 Detailed Description

Contains implementations of `cc_ht_map_`'s find related functions, when the hash value is stored.

Definition in file [cc\\_hash\\_table\\_map\\_/find\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.218 find\_store\_hash\_fn\_imps.hpp File Reference

### 5.218.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions, when the hash value is stored.

Definition in file [gp\\_hash\\_table\\_map\\_/find\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.219 for\_each.h File Reference

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename `_Iter`, typename `_UserOp`, typename `_Functionality`, typename `_Red`, typename `_Result` >  
`_UserOp` [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access](#) (`_Iter` `__begin`, `_Iter` `__end`, `_UserOp` `__user_op`, `_Functionality` & `__functionality`, `_Red` `__reduction`, `_Result` `__reduction_start`, `_Result` & `__output`, typename `std::iterator_traits<_Iter>::difference_type` `__bound`, `_Parallelism` `__parallelism_tag`)

### 5.219.1 Detailed Description

Main interface for embarrassingly parallel functions. The explicit implementation are in other header files, like `workstealing.h`, `par_loop.h`, `omp_loop.h`, and `omp_loop_static.h`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each.h](#).

5.220 `for_each_selectors.h` File Reference

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_binop\\_reduct<\\_BinOp>](#)  
*General reduction, using a binary operator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_selector<\\_It>](#)  
*std::accumulate() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_difference\\_selector<\\_It>](#)  
*Selector that returns the difference between two adjacent \_\_elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_if\\_selector<\\_It, \\_Diff>](#)  
*std::count\_if() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_selector<\\_It, \\_Diff>](#)  
*std::count() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_fill\\_selector<\\_It>](#)  
*std::fill() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector<\\_It>](#)  
*std::for\_each() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generate\\_selector<\\_It>](#)  
*std::generate() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector<\\_It>](#)  
*Generic \_\_selector for embarrassingly parallel functions.*
- struct [\\_\\_gnu\\_parallel::\\_\\_identity\\_selector<\\_It>](#)  
*Selector that just returns the passed iterator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector<\\_It, \\_It2, \\_Tp>](#)  
*std::inner\_product() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_max\\_element\\_reduct<\\_Compare, \\_It>](#)  
*Reduction for finding the maximum element, using a comparator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_min\\_element\\_reduct<\\_Compare, \\_It>](#)  
*Reduction for finding the maximum element, using a comparator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_replace\\_if\\_selector<\\_It, \\_Op, \\_Tp>](#)  
*std::replace() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_replace\\_selector<\\_It, \\_Tp>](#)  
*std::replace() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector<\\_It>](#)  
*std::transform() \_\_selector, one input sequence variant.*
- struct [\\_\\_gnu\\_parallel::\\_\\_transform2\\_selector<\\_It>](#)  
*std::transform() \_\_selector, two input sequences variant.*
- struct [\\_\\_gnu\\_parallel::\\_\\_DummyReduct](#)  
*Reduction function doing nothing.*
- struct [\\_\\_gnu\\_parallel::\\_\\_Nothing](#)  
*Functor doing nothing.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## 5.220.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each\\_selectors.h](#).

5.221 `formatter.h` File Reference

## Namespaces

- namespace [\\_\\_gnu\\_debug](#)

## Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,  
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move_`-  
`assign`,  
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs` }

## Functions

- `template<typename _Iterator>`  
`bool __gnu_debug::__check_singular (_Iterator &)`

## 5.221.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [formatter.h](#).

5.222 `forward_list` File Reference

## Macros

- `#define _GLIBCXX_FORWARD_LIST`

## 5.222.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list](#).

## 5.223 forward\_list File Reference

## Classes

- class [std::\\_\\_debug::forward\\_list](#)< \_Tp, \_Alloc >  
*Class std::forward\_list with safety/checking/debug instrumentation.*

## Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Macros

- #define [\\_GLIBCXX\\_DEBUG\\_FORWARD\\_LIST](#)

## Functions

- template<typename \_Tp, typename \_Alloc >  
bool [std::\\_\\_debug::operator!=](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::\\_\\_debug::operator<](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::\\_\\_debug::operator<=](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::\\_\\_debug::operator==](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::\\_\\_debug::operator>](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::\\_\\_debug::operator>=](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
void [std::\\_\\_debug::swap](#) (forward\_list< \_Tp, \_Alloc > &\_\_lx, forward\_list< \_Tp, \_Alloc > &\_\_ly)

## 5.223.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/forward\\_list](#).

## 5.224 forward\_list File Reference

### Classes

- class [std::\\_\\_profile::forward\\_list< \\_Tp, \\_Alloc >](#)  
*Class std::forward\_list wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Macros

- `#define \_GLIBCXX\_PROFILE\_FORWARD\_LIST`

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_profile::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_profile::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_profile::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_profile::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_profile::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_profile::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::\_\_profile::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`

#### 5.224.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [profile/forward\\_list](#).

## 5.225 forward\_list.h File Reference

### Classes

- struct [std::\\_Fwd\\_list\\_base< \\_Tp, \\_Alloc >](#)  
*Base class for forward\_list.*

- struct [std::\\_Fwd\\_list\\_const\\_iterator<\\_Tp>](#)  
*A forward\_list::const\_iterator.*
- struct [std::\\_Fwd\\_list\\_iterator<\\_Tp>](#)  
*A forward\_list::iterator.*
- struct [std::\\_Fwd\\_list\\_node<\\_Tp>](#)  
*A helper node class for forward\_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.*
- struct [std::\\_Fwd\\_list\\_node\\_base](#)  
*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*
- class [std::forward\\_list<\\_Tp, \\_Alloc>](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

## Namespaces

- namespace [std](#)

## Functions

- template<typename \_Tp>  
bool [std::operator!=](#) (const \_Fwd\_list\_iterator<\_Tp> &\_\_x, const \_Fwd\_list\_const\_iterator<\_Tp> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator!=](#) (const forward\_list<\_Tp, \_Alloc> &\_\_lx, const forward\_list<\_Tp, \_Alloc> &\_\_ly)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator<](#) (const forward\_list<\_Tp, \_Alloc> &\_\_lx, const forward\_list<\_Tp, \_Alloc> &\_\_ly)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator<=](#) (const forward\_list<\_Tp, \_Alloc> &\_\_lx, const forward\_list<\_Tp, \_Alloc> &\_\_ly)
- template<typename \_Tp>  
bool [std::operator==](#) (const \_Fwd\_list\_iterator<\_Tp> &\_\_x, const \_Fwd\_list\_const\_iterator<\_Tp> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator==](#) (const forward\_list<\_Tp, \_Alloc> &\_\_lx, const forward\_list<\_Tp, \_Alloc> &\_\_ly)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator>](#) (const forward\_list<\_Tp, \_Alloc> &\_\_lx, const forward\_list<\_Tp, \_Alloc> &\_\_ly)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator>=](#) (const forward\_list<\_Tp, \_Alloc> &\_\_lx, const forward\_list<\_Tp, \_Alloc> &\_\_ly)
- template<typename \_Tp, typename \_Alloc>  
void [std::swap](#) (forward\_list<\_Tp, \_Alloc> &\_\_lx, forward\_list<\_Tp, \_Alloc> &\_\_ly)

### 5.225.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

Definition in file [forward\\_list.h](#).

## 5.226 forward\_list.tcc File Reference

### Namespaces

- namespace [std](#)

**Macros**

- `#define _FORWARD_LIST_TCC`

**Functions**

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

**5.226.1 Detailed Description**

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

Definition in file [forward\\_list.tcc](#).

**5.227 [fstream File Reference](#)****Classes**

- class [std::basic\\_filebuf](#)< [\\_CharT](#), [\\_Traits](#) >  
*The actual work of input and output (for files).*
- class [std::basic\\_fstream](#)< [\\_CharT](#), [\\_Traits](#) >  
*Controlling input and output for files.*
- class [std::basic\\_ifstream](#)< [\\_CharT](#), [\\_Traits](#) >  
*Controlling input for files.*
- class [std::basic\\_ofstream](#)< [\\_CharT](#), [\\_Traits](#) >  
*Controlling output for files.*

**Namespaces**

- namespace [std](#)

**Macros**

- `#define _GLIBCXX_FSTREAM`

**5.227.1 Detailed Description**

This is a Standard C++ Library header.

Definition in file [fstream](#).

**5.228 [fstream.tcc File Reference](#)****Namespaces**

- namespace [std](#)

## Macros

- `#define _FSTREAM_TCC`

## 5.228.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<fstream>`.

Definition in file [fstream.tcc](#).

5.229 `functexcept.h` File Reference

## Namespaces

- namespace [std](#)

## Functions

- void **std::\_\_throw\_bad\_alloc** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_cast** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_exception** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_function\_call** () `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_typeid** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_domain\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_future\_error** (int) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_invalid\_argument** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_ios\_failure** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_length\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_logic\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_out\_of\_range** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_overflow\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_range\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_runtime\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_system\_error** (int) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_underflow\_error** (const char \*) `__attribute__((__noreturn__))`

## 5.229.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

Definition in file [functexcept.h](#).

## 5.230 functional File Reference

## Classes

- struct [\\_Bind< \\_Signature >](#)  
*Type of the function object returned from bind().*
- struct [\\_Bind\\_result< \\_Result, \\_Signature >](#)  
*Type of the function object returned from bind<R>().*
- class [\\_Mu< \\_Arg, \\_IsBindExp, \\_IsPlaceholder >](#)
- struct [\\_Reference\\_wrapper\\_base\\_impl< \\_Unary, \\_Binary, \\_Tp >](#)
- struct [std::\\_is\\_location\\_invariant< \\_Tp >](#)
- struct [std::\\_Derives\\_from\\_binary\\_function< \\_Tp >](#)  
*Determines if the type \_Tp derives from binary\_function.*
- struct [std::\\_Derives\\_from\\_unary\\_function< \\_Tp >](#)  
*Determines if the type \_Tp derives from unary\_function.*
- class [std::\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [std::\\_Maybe\\_get\\_result\\_type< \\_Has\\_result\\_type, \\_Functor >](#)  
*If we have found a result\_type, extract it.*
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_ArgTypes >](#)
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1 >](#)  
*Derives from unary\_function, as appropriate.*
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1, \\_T2 >](#)  
*Derives from binary\_function, as appropriate.*
- struct [std::\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp >](#)
- struct [std::\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp \\_Class::\\* >](#)
- class [std::\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Implementation of mem\_fn for const member function pointers.*
- class [std::\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Implementation of mem\_fn for const volatile member function pointers.*
- class [std::\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Implementation of mem\_fn for volatile member function pointers.*
- class [std::\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\)>](#)  
*Implementation of mem\_fn for member function pointers.*
- class [std::\\_Mu< \\_Arg, false, false >](#)
- class [std::\\_Mu< \\_Arg, false, true >](#)
- class [std::\\_Mu< \\_Arg, true, false >](#)
- class [std::\\_Mu< reference\\_wrapper< \\_Tp >, false, false >](#)
- struct [std::\\_Placeholder< \\_Num >](#)  
*The type of placeholder objects defined by libstdc++.*
- struct [std::\\_Reference\\_wrapper\\_base< \\_Tp >](#)
- struct [std::\\_Safe\\_tuple\\_element< \\_\\_i, \\_Tuple >](#)
- struct [std::\\_Safe\\_tuple\\_element\\_impl< \\_\\_i, \\_Tuple, \\_IsSafe >](#)
- struct [std::\\_Safe\\_tuple\\_element\\_impl< \\_\\_i, \\_Tuple, false >](#)
- struct [std::\\_Weak\\_result\\_type< \\_Functor >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl< \\_Functor >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl< \\_Res\(&\)\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function reference.*
- struct [std::\\_Weak\\_result\\_type\\_impl< \\_Res\(\\*\)\(\\_ArgTypes...\)>](#)

- Retrieve the result type for a function pointer.*

  - struct `std::Weak_result_type_impl<_Res(_ArgTypes...)>`
- Retrieve the result type for a function type.*

  - struct `std::Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`
- Retrieve result type for a const member function pointer.*

  - struct `std::Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile >`
- Retrieve result type for a const volatile member function pointer.*

  - struct `std::Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) volatile >`
- Retrieve result type for a volatile member function pointer.*

  - struct `std::Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...)>`
- Retrieve result type for a member function pointer.*

  - class `std::bad_function_call`
- Exception class thrown when class template function's operator() is called with an empty target.*

  - class `std::function<_Res(_ArgTypes...)>`
- Primary class template for std::function.*  
*Polymorphic function wrapper.*

  - struct `std::is_bind_expression<_Tp >`
- Determines if the given type \_Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].*

  - struct `std::is_bind_expression<_Bind<_Signature > >`
- Class template \_Bind is always a bind expression.*

  - struct `std::is_bind_expression<_Bind_result<_Result, _Signature > >`
- Class template \_Bind\_result is always a bind expression.*

  - struct `std::is_bind_expression<const _Bind<_Signature > >`
- Class template \_Bind is always a bind expression.*

  - struct `std::is_bind_expression<const _Bind_result<_Result, _Signature > >`
- Class template \_Bind\_result is always a bind expression.*

  - struct `std::is_bind_expression<const volatile _Bind<_Signature > >`
- Class template \_Bind is always a bind expression.*

  - struct `std::is_bind_expression<const volatile _Bind_result<_Result, _Signature > >`
- Class template \_Bind\_result is always a bind expression.*

  - struct `std::is_bind_expression<volatile _Bind<_Signature > >`
- Class template \_Bind is always a bind expression.*

  - struct `std::is_bind_expression<volatile _Bind_result<_Result, _Signature > >`
- Class template \_Bind\_result is always a bind expression.*

  - struct `std::is_placeholder<_Tp >`
- Determines if the given type \_Tp is a placeholder in a bind() expression and, if so, which placeholder it is. [TR1 3.6.2].*

  - struct `std::is_placeholder<_Placeholder<_Num > >`
- Primary class template for reference\_wrapper.*

  - class `std::reference_wrapper<_Tp >`

## Namespaces

- namespace `std`
- namespace `std::placeholders`

## Macros

- `#define _GLIBCXX_FUNCTIONAL`

## Typedefs

- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>  
using std::__is_socketlike = __or_< is_integral< _Tp2 >, is_enum< _Tp2 >>`
- `template<typename _Tp1, typename _Tp2 >  
using std::__NotSame = __not_< is_same< typename std::decay< _Tp1 >::type, typename std::decay< _Tp2 >::type >>`

## Enumerations

- `enum _Manager_operation { __get_type_info, __get_func_ptr, __clone_func_ptr, __destroy_func_ptr }`

## Functions

- `template<typename _Callable, typename... _Args>  
_Bind_simple_helper< _Callable,  
_Args...>::type std::__bind_simple ( _Callable &&__callable, _Args &&...__args)`
- `template<typename _Functor >  
_Functor & std::__callable_func_ptr ( _Functor &__f)`
- `template<typename _Member, typename _Class >  
_Mem_fn< _Member _Class::* > std::__callable_func_ptr ( _Member _Class::*__p)`
- `template<typename _Member, typename _Class >  
_Mem_fn< _Member _Class::* > std::__callable_func_ptr ( _Member _Class::*const &__p)`
- `template<typename _Member, typename _Class >  
_Mem_fn< _Member _Class::* > std::__callable_func_ptr ( _Member _Class::*volatile &__p)`
- `template<typename _Member, typename _Class >  
_Mem_fn< _Member _Class::* > std::__callable_func_ptr ( _Member _Class::*const volatile &__p)`
- `template<typename _Functor, typename... _Args>  
enable_if< (is_member_pointer  
< _Functor >::value  
&&is_function< _Functor >  
::value &&is_function  
< typename remove_pointer  
< _Functor >::type >::value),  
typename result_of< _Functor(_Args &&...)>  
::type >::type std::__invoke ( _Functor &__f, _Args &&...__args)`
- `template<typename _Functor, typename... _Args>  
enable_if< (is_pointer  
< _Functor >::value  
&&is_function< typename  
remove_pointer< _Functor >  
::type >::value), typename  
result_of< _Functor(_Args &&...)>  
::type >::type std::__invoke ( _Functor __f, _Args &&...__args)`
- `template<std::size_t _Ind, typename... _Tp>  
auto std::__volget (volatile tuple< _Tp...> &__tuple) -> typename tuple_element< _Ind, tuple< _Tp...>>::type  
volatile &`

- `template<std::size_t _Ind, typename... _Tp>`  
`auto std::__volget (const volatile tuple< _Tp...> &__tuple) -> typename tuple_element< _Ind, tuple< _Tp...>>::type const volatile &`
- `template<typename _Func, typename... _BoundArgs>`  
`_Bind_helper< __is_socketlike`  
`< _Func >::value, _Func,`  
`_BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&...__args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`_Bindres_helper< _Result,`  
`_Func, _BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&...__args)`
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::* __pm) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool std::operator!= (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool std::operator!= (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `template<typename _Res, typename... _Args>`  
`void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)`
  
- `template<typename _Tp >`  
`reference_wrapper< _Tp > std::ref (_Tp &__t) noexcept`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (const _Tp &__t) noexcept`
- `template<typename _Tp >`  
`void std::ref (const _Tp &&)=delete`
- `template<typename _Tp >`  
`void std::cref (const _Tp &&)=delete`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > std::ref (reference_wrapper< _Tp > __t) noexcept`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (reference_wrapper< _Tp > __t) noexcept`

## Variables

- `const _Placeholder< 1 > std::placeholders::_1`
- `const _Placeholder< 10 > std::placeholders::_10`
- `const _Placeholder< 11 > std::placeholders::_11`
- `const _Placeholder< 12 > std::placeholders::_12`
- `const _Placeholder< 13 > std::placeholders::_13`
- `const _Placeholder< 14 > std::placeholders::_14`
- `const _Placeholder< 15 > std::placeholders::_15`
- `const _Placeholder< 16 > std::placeholders::_16`
- `const _Placeholder< 17 > std::placeholders::_17`
- `const _Placeholder< 18 > std::placeholders::_18`
- `const _Placeholder< 19 > std::placeholders::_19`
- `const _Placeholder< 2 > std::placeholders::_2`
- `const _Placeholder< 20 > std::placeholders::_20`

- `const _Placeholder< 21 > std::placeholders::_21`
- `const _Placeholder< 22 > std::placeholders::_22`
- `const _Placeholder< 23 > std::placeholders::_23`
- `const _Placeholder< 24 > std::placeholders::_24`
- `const _Placeholder< 25 > std::placeholders::_25`
- `const _Placeholder< 26 > std::placeholders::_26`
- `const _Placeholder< 27 > std::placeholders::_27`
- `const _Placeholder< 28 > std::placeholders::_28`
- `const _Placeholder< 29 > std::placeholders::_29`
- `const _Placeholder< 3 > std::placeholders::_3`
- `const _Placeholder< 4 > std::placeholders::_4`
- `const _Placeholder< 5 > std::placeholders::_5`
- `const _Placeholder< 6 > std::placeholders::_6`
- `const _Placeholder< 7 > std::placeholders::_7`
- `const _Placeholder< 8 > std::placeholders::_8`
- `const _Placeholder< 9 > std::placeholders::_9`

### 5.230.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [functional](#).

## 5.231 functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose< \\_Operation1, \\_Operation2, \\_Operation3 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun< \\_Result, \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun< \\_Result, \\_Argument >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun< \\_Result >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project1st< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project2nd< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::select1st< \\_Pair >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::select2nd< \\_Pair >](#)  
*An SGI extension .*
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose< \\_Operation1, \\_Operation2 >](#)  
*An SGI extension .*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_FUNCTIONAL`

## Functions

- `template<class _Operation1 , class _Operation2 >`  
`unary_compose< _Operation1,`  
`_Operation2 > \_\_gnu\_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`  
`binary_compose< _Operation1,`  
`_Operation2, _Operation3 > \_\_gnu\_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const`  
`_Operation3 &__fn3)`
- `template<class _Result >`  
`constant_void_fun< _Result > \_\_gnu\_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun< _Result,`  
`_Result > \_\_gnu\_cxx::constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant_binary_fun< _Result,`  
`_Result, _Result > \_\_gnu\_cxx::constant2 (const _Result &__val)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::plus< _Tp >)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::multiplies< _Tp >)`
- `template<class _Ret , class _Tp , class _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1 (_Ret(_Tp::*)(_Arg))`
- `template<class _Ret , class _Tp , class _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1\_ref (_Ret(_Tp::*)(_Arg))`

## 5.231.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/functional](#).

## 5.232 functional\_hash.h File Reference

## Classes

- struct [hash< \\_Tp >](#)  
*Primary class template hash.*
- struct [std::hash< \\_Tp \\* >](#)  
*Partial specializations for pointer types.*
- struct [std::hash< bool >](#)  
*Explicit specialization for bool.*

- struct `std::hash< char >`  
*Explicit specialization for char.*
- struct `std::hash< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `std::hash< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `std::hash< double >`  
*Specialization for double.*
- struct `std::hash< float >`  
*Specialization for float.*
- struct `std::hash< int >`  
*Explicit specialization for int.*
- struct `std::hash< long >`  
*Explicit specialization for long.*
- struct `std::hash< long double >`  
*Specialization for long double.*
- struct `std::hash< long long >`  
*Explicit specialization for long long.*
- struct `std::hash< short >`  
*Explicit specialization for short.*
- struct `std::hash< signed char >`  
*Explicit specialization for signed char.*
- struct `std::hash< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `std::hash< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `std::hash< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `std::hash< unsigned long long >`  
*Explicit specialization for unsigned long long.*
- struct `std::hash< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `std::hash< wchar_t >`  
*Explicit specialization for wchar\_t.*

## Namespaces

- namespace `std`

## Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

### 5.232.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file `functional_hash.h`.

## 5.233 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _Iterator >  
_Siter_base< _Iterator >  
::iterator_type \_\_gnu\_debug::\_\_base (_Iterator __it)`
- `template<typename _Iterator >  
bool \_\_gnu\_debug::\_\_check\_dereferenceable (_Iterator &)`
- `template<typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Tp * __ptr)`
- `template<typename _Iterator, typename _Sequence >  
bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > & __x)`
- `template<typename _ForwardIterator, typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp  
& __value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp  
& __value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_lower\_aux (_ForwardIterator __first, _ForwardIterator __last, const  
_Tp & __value, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_lower\_aux (const _Safe_iterator< _Iterator, _Sequence > & __first,  
const _Safe_iterator< _Iterator, _Sequence > & __last, const _Tp & __value, std::random\_access\_iterator\_tag  
__tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_lower\_aux (_ForwardIterator __first, _ForwardIterator __last, const  
_Tp & __value, _Pred __pred, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Tp, typename _Pred >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_lower\_aux (const _Safe_iterator< _Iterator, _Sequence > & __first,  
const _Safe_iterator< _Iterator, _Sequence > & __last, const _Tp & __value, _Pred __pred, std::random\_access-  
iterator\_tag __tag)`
- `template<typename _ForwardIterator, typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp  
& __value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp  
& __value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_upper\_aux (_ForwardIterator __first, _ForwardIterator __last, const  
_Tp & __value, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Tp >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_upper\_aux (const _Safe_iterator< _Iterator, _Sequence > & __first,  
const _Safe_iterator< _Iterator, _Sequence > & __last, const _Tp & __value, std::random\_access\_iterator\_tag  
__tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >  
bool \_\_gnu\_debug::\_\_check\_partitioned\_upper\_aux (_ForwardIterator __first, _ForwardIterator __last, const  
_Tp & __value, _Pred __pred, std::forward\_iterator\_tag)`

- `template<typename _Iterator, typename _Sequence, typename _Tp, typename _Pred >`  
`bool __gnu_debug::__check_partitioned_upper_aux (const _Safe_iterator< _Iterator, _Sequence > &__first,`  
`const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, _Pred __pred, std::random\_access\_iterator\_tag __tag)`
- `template<typename _Iterator >`  
`bool __gnu_debug::__check_singular (_Iterator &)`
- `template<typename _Tp >`  
`bool __gnu_debug::__check_singular (const _Tp * __ptr)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::__check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `bool __gnu_debug::__check_singular_aux (const void *)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::__check_sorted_aux (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, std::random\_access\_iterator\_tag __tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, _Predicate __pred, std::random\_access\_iterator\_tag __tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::true\_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::true\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false\_type)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * __gnu_debug::__check_string (const _CharT * __s, const _Integer &__n __attribute__((__unused__)))`

- `template<typename _CharT >`  
`const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT * __s)`
- `template<typename _InputIterator >`  
`_InputIterator \_\_gnu\_debug::\_\_check\_valid\_range (const _InputIterator & __first, const _InputIterator & __last`  
`__attribute__((__unused__)))`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _InputIterator & __first, const _InputIterator & __last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence > & __first, const _Safe_iterator<`  
`_Iterator, _Sequence > & __last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_local_iterator< _Iterator, _Sequence > & __first, const _Safe_`  
`local_iterator< _Iterator, _Sequence > & __last)`
- `template<typename _Integral >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _InputIterator & __first, const _InputIterator & __last, std::__false_`  
`type)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux2 (const _RandomAccessIterator & __first, const _RandomAccessIterator`  
`& __last, std::random_access_iterator_tag)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux2 (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`

#### 5.233.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [functions.h](#).

## 5.234 future File Reference

### Classes

- class [std::future\\_error](#)  
*Exception type thrown by futures.*
- struct [std::is\\_error\\_code\\_enum< future\\_errc >](#)  
*Specialization.*

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_FUTURE`

## Enumerations

- enum [std::future\\_errc](#)
- enum [std::future\\_status](#)
- enum [std::launch](#)

## Functions

- template<typename \_Fn, typename... \_Args>  
future< typename [result\\_of](#)  
< \_Fn(\_Args...) >::type > **std::async** (launch \_\_policy, \_Fn &&\_\_fn, \_Args &&...\_\_args)
- template<typename \_Fn, typename... \_Args>  
future< typename [result\\_of](#)  
< \_Fn(\_Args...) >::type > **std::async** (\_Fn &&\_\_fn, \_Args &&...\_\_args)
- const error\_category & [std::future\\_category](#) () noexcept
- error\_code [std::make\\_error\\_code](#) (future\_errc \_\_errc) noexcept
- error\_condition [std::make\\_error\\_condition](#) (future\_errc \_\_errc) noexcept
- constexpr launch **std::operator&** (launch \_\_x, launch \_\_y)
- launch & **std::operator&=** (launch &\_\_x, launch \_\_y)
- constexpr launch **std::operator^** (launch \_\_x, launch \_\_y)
- launch & **std::operator^=** (launch &\_\_x, launch \_\_y)
- constexpr launch **std::operator|** (launch \_\_x, launch \_\_y)
- launch & **std::operator|=** (launch &\_\_x, launch \_\_y)
- constexpr launch **std::operator~** (launch \_\_x)

## 5.234.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [future](#).

## 5.235 gp\_ht\_map.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_GEN\_POS**
- #define **PB\_DS\_GP\_HASH\_NAME**
- #define **PB\_DS\_GP\_HASH\_TRAITS\_BASE**
- #define **PB\_DS\_HASH\_EQ\_FN\_C\_DEC**
- #define **PB\_DS\_RANGED\_PROBE\_FN\_C\_DEC**

## Variables

- `empty_entry_status`
- `erased_entry_status`
- `valid_entry_status`

## 5.235.1 Detailed Description

Contains an implementation class for general probing hash.

Definition in file [gp\\_ht\\_map\\_.hpp](#).

5.236 `gslice.h` File Reference

## Classes

- class [std::gslice](#)  
*Class defining multi-dimensional subset of an array.*

## Namespaces

- namespace [std](#)

## 5.236.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [gslice.h](#).

5.237 `gslice_array.h` File Reference

## Classes

- class [std::gslice\\_array<\\_Tp>](#)  
*Reference to multi-dimensional subset of an array.*

## Namespaces

- namespace [std](#)

## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 5.237.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [gslice\\_array.h](#).

5.238 `hash_bytes.h` File Reference

## Namespaces

- namespace [std](#)

## Functions

- `size_t std::Fnv_hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)
- `size_t std::Hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)

## 5.238.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [hash\\_bytes.h](#).

5.239 `hash_eq_fn.hpp` File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, false >](#)  
*Specialization 1 - The client requests that hash values not be stored.*
- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, true >](#)  
*Specialization 2 - The client requests that hash values be stored.*
- struct [hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, Store\\_Hash >](#)  
*Primary template.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.239.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

Definition in file [hash\\_eq\\_fn.hpp](#).

## 5.240 hash\_exponential\_size\_policy\_imp.hpp File Reference

### 5.240.1 Detailed Description

Contains a resize size policy implementation.

Definition in file [hash\\_exponential\\_size\\_policy\\_imp.hpp](#).

## 5.241 hash\_fun.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `size_t __gnu_cxx::__stl_hash_string` (const char \*\_\_s)

### 5.241.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

Definition in file [hash\\_fun.h](#).

## 5.242 hash\_load\_check\_resize\_trigger\_imp.hpp File Reference

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_ASSERT_VALID(X)`

### 5.242.1 Detailed Description

Contains a resize trigger implementation.

Definition in file [hash\\_load\\_check\\_resize\\_trigger\\_imp.hpp](#).

## 5.243 hash\_load\_check\_resize\_trigger\_size\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, true >](#)  
*Specializations.*
- class [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, Hold\\_Size >](#)  
*Primary template.*

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.243.1 Detailed Description

Contains an base holding size for some resize policies.

Definition in file [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#).

## 5.244 hash\_map File Reference

## Classes

- class [\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Macros

- `#define _BACKWARD_HASH_MAP`

## Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
void __gnu_cxx::swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
void __gnu_cxx::swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`

## 5.244.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

Definition in file [hash\\_map](#).

## 5.245 hash\_policy.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)  
A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.
- class [\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#)  
A mask range-hashing class (uses a bitmask).
- class [\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#)  
A mod range-hashing class (uses the modulo function).
- class [\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#)  
A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).
- class [\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)  
A resize trigger policy based on a load check. It keeps the load factor between some load factors load\_min and load\_max.
- class [\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#)  
A size policy whose sequence of sizes form a nearly-exponential sequence of primes.
- class [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#)  
A resize policy which delegates operations to size and trigger policies.
- class [\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#)  
A probe sequence policy using fixed increments.
- class [\\_\\_gnu\\_pbds::quadratic\\_probe\\_fn< Size\\_Type >](#)  
A probe sequence policy using square increments.

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

## Enumerations

- enum { **num\_distinct\_sizes\_32\_bit**, **num\_distinct\_sizes\_64\_bit**, **num\_distinct\_sizes** }

## Variables

- static const std::size\_t **\_\_gnu\_pbds::detail::g\_a\_sizes** [num\_distinct\_sizes\_64\_bit]

## 5.245.1 Detailed Description

Contains hash-related policies.

Definition in file [hash\\_policy.hpp](#).

## 5.246 hash\_prime\_size\_policy\_imp.hpp File Reference

## Enumerations

- enum { **num\_distinct\_sizes\_32\_bit**, **num\_distinct\_sizes\_64\_bit**, **num\_distinct\_sizes** }

## Variables

- static const std::size\_t **detail::g\_a\_sizes** [num\_distinct\_sizes\_64\_bit]

## 5.246.1 Detailed Description

Contains a resize size policy implementation.

Definition in file [hash\\_prime\\_size\\_policy\\_imp.hpp](#).

## 5.247 hash\_set File Reference

## Classes

- class [\\_\\_gnu\\_cxx::hash\\_multiset](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc >
- class [\\_\\_gnu\\_cxx::hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc >

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Macros

- #define **\_BACKWARD\_HASH\_SET**

## Functions

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

## 5.247.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_set](#).

## 5.248 hash\_standard\_resize\_policy\_imp.hpp File Reference

## 5.248.1 Detailed Description

Contains a resize policy implementation.

Definition in file [hash\\_standard\\_resize\\_policy\\_imp.hpp](#).

## 5.249 hashtable.h File Reference

## Classes

- class [std::\\_Hashtable< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits >](#)

## Namespaces

- namespace [std](#)

## Typedefs

- `template<typename _Tp, typename _Hash >`  
`using std::__cache_default = __not_< __and_< __is_fast_hash< _Hash >, is_default_constructible< _Hash >, is_copy_assignable< _Hash >, __detail::__is_noexcept_hash< _Tp, _Hash >>>`

## 5.249.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

Definition in file [bits/hashtable.h](#).

5.250 `hashtable.h` File Reference

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Enumerations

- enum { **\_S\_num\_primes** }

## Functions

- unsigned long [\\_\\_gnu\\_cxx::\\_\\_stl\\_next\\_prime](#) (unsigned long \_\_n)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All >  
bool [\\_\\_gnu\\_cxx::operator!=](#) (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All >  
bool [\\_\\_gnu\\_cxx::operator==](#) (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Extract, class \_EqKey, class \_All >  
void [\\_\\_gnu\\_cxx::swap](#) (hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht1, hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht2)

## 5.250.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

Definition in file [backward/hashtable.h](#).

5.251 `hashtable_policy.h` File Reference

## Classes

- struct [\\_Equal\\_helper](#)< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, \_\_cache\_hash\_code >
- struct [\\_Equality](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >
- struct [\\_Hash\\_code\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct [\\_Hash\\_node](#)< \_Value, \_Cache\_hash\_code >
- struct [\\_Hashtable\\_ebo\\_helper](#)< \_Nm, \_Tp, \_\_use\_ebo >
- struct [\\_Insert](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Constant\_iterators, \_Unique\_keys >
- struct [\\_Local\\_iterator\\_base](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct [\\_Rehash\\_base](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

- struct `std::__detail::_Before_begin<_NodeAlloc>`
- struct `std::__detail::_Default_ranged_hash`  
*Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that  $h(k, N) = h2(h1(k), N)$ , but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.*
- struct `std::__detail::_Equal_helper<_Key, _Value, _ExtractKey, _Equal, _HashCodeType, false>`  
*Specialization.*
- struct `std::__detail::_Equal_helper<_Key, _Value, _ExtractKey, _Equal, _HashCodeType, true>`  
*Specialization.*
- struct `std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false>`  
*Specialization.*
- struct `std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true>`  
*Specialization.*
- struct `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false>`  
*Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.*
- struct `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true>`  
*Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.*
- struct `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false>`  
*Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.*
- struct `std::__detail::_Hash_node<_Value, false>`
- struct `std::__detail::_Hash_node<_Value, true>`
- struct `std::__detail::_Hash_node_base`
- struct `std::__detail::_Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits>`
- struct `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false>`  
*Specialization not using EBO.*
- struct `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, true>`  
*Specialization using EBO.*
- struct `std::__detail::_Hashtable_traits<_Cache_hash_code, _Constant_iterators, _Unique_keys>`
- struct `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys>`  
*Specialization.*
- struct `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false>`  
*Specialization.*
- struct `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true>`  
*Specialization.*
- struct `std::__detail::_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits>`
- struct `std::__detail::_Local_const_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache>`  
*local const iterators*
- struct `std::__detail::_Local_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache>`

*local iterators*

- struct `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`

*Specialization.*

- struct `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`

*Specialization.*

- struct `std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- struct `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`

*Partial specialization, \_\_unique\_keys set to false.*

- struct `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`

*Partial specialization, \_\_unique\_keys set to true.*

- struct `std::__detail::Mod_range_hashing`

*Default range hashing function: use division to fold a large number into the range [0, N).*

- struct `std::__detail::Node_const_iterator< _Value, __constant_iterators, __cache >`

*Node const\_iterators, used to iterate through all the hashtable.*

- struct `std::__detail::Node_iterator< _Value, __constant_iterators, __cache >`

*Node iterators, used to iterate through all the hashtable.*

- struct `std::__detail::Node_iterator_base< _Value, _Cache_hash_code >`

*Base class for node iterators.*

- struct `std::__detail::Prime_rehash_policy`

*Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.*

- struct `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >`

*Specialization.*

- class `std::Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

## Namespaces

- namespace `std`
- namespace `std::__detail`

## Functions

- template<class \_Iterator >  
std::iterator\_traits  
< \_Iterator >::difference\_type `std::__detail::__distance_fw` (\_Iterator \_\_first, \_Iterator \_\_last, `std::input_iterator_tag`)
- template<class \_Iterator >  
std::iterator\_traits  
< \_Iterator >::difference\_type `std::__detail::__distance_fw` (\_Iterator \_\_first, \_Iterator \_\_last, `std::forward_iterator_tag`)
- template<class \_Iterator >  
std::iterator\_traits  
< \_Iterator >::difference\_type `std::__detail::__distance_fw` (\_Iterator \_\_first, \_Iterator \_\_last)
- template<typename \_Value, bool \_Cache\_hash\_code>  
bool `std::__detail::operator!=` (const \_Node\_iterator\_base< \_Value, \_Cache\_hash\_code > &\_\_x, const \_Node\_iterator\_base< \_Value, \_Cache\_hash\_code > &\_\_y)

- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool std::__detail::operator!= (const \_Local\_iterator\_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __-`  
`cache > &__x, const \_Local\_iterator\_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool std::__detail::operator== (const \_Node\_iterator\_base< _Value, _Cache_hash_code > &__x, const \_Node\_iterator\_base< _Value, _Cache_hash_code > &__y)`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool std::__detail::operator== (const \_Local\_iterator\_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __-`  
`cache > &__x, const \_Local\_iterator\_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`

#### 5.251.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

Definition in file [hashtable\\_policy.h](#).

## 5.252 indirect\_array.h File Reference

### Classes

- class [std::indirect\\_array< \\_Tp >](#)  
*Reference to arbitrary subset of an array.*

### Namespaces

- namespace [std](#)

### Macros

- `#define \_DEFINE\_VALARRAY\_OPERATOR(_Op, _Name)`

#### 5.252.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [indirect\\_array.h](#).

## 5.253 info\_fn\_imps.hpp File Reference

#### 5.253.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary\\_heap\\_/info\\_fn\\_imps.hpp](#).

## 5.254 `info_fn_imps.hpp` File Reference

### 5.254.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin\\_search\\_tree\\_/info\\_fn\\_imps.hpp](#).

## 5.255 `info_fn_imps.hpp` File Reference

### 5.255.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container info related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/info\\_fn\\_imps.hpp](#).

## 5.256 `info_fn_imps.hpp` File Reference

### 5.256.1 Detailed Description

Contains implementations of `gp_ht_map_`'s entire container info related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/info\\_fn\\_imps.hpp](#).

## 5.257 `info_fn_imps.hpp` File Reference

### 5.257.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/info\\_fn\\_imps.hpp](#).

## 5.258 `info_fn_imps.hpp` File Reference

### 5.258.1 Detailed Description

Contains implementations of `lu_map_`.

Definition in file [list\\_update\\_map\\_/info\\_fn\\_imps.hpp](#).

## 5.259 `info_fn_imps.hpp` File Reference

### 5.259.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov\\_tree\\_map\\_/info\\_fn\\_imps.hpp](#).

## 5.260 `info_fn_imps.hpp` File Reference

#### 5.260.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [pat\\_trie\\_/info\\_fn\\_imps.hpp](#).

### 5.261 `info_fn_imps.hpp` File Reference

#### 5.261.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb\\_tree\\_map\\_/info\\_fn\\_imps.hpp](#).

### 5.262 `info_fn_imps.hpp` File Reference

#### 5.262.1 Detailed Description

Contains an implementation.

Definition in file [splay\\_tree\\_/info\\_fn\\_imps.hpp](#).

### 5.263 `initializer_list` File Reference

#### Classes

- class [std::initializer\\_list<\\_E>](#)  
*initializer\_list*

#### Namespaces

- namespace [std](#)

#### Functions

- `template<class _Tp>`  
`constexpr const _Tp * std::begin (initializer_list<_Tp> __ils) noexcept`
- `template<class _Tp>`  
`constexpr const _Tp * std::end (initializer_list<_Tp> __ils) noexcept`

#### 5.263.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [initializer\\_list](#).

### 5.264 `insert_fn_imps.hpp` File Reference

#### 5.264.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary\\_heap\\_/insert\\_fn\\_imps.hpp](#).

### 5.265 `insert_fn_imps.hpp` File Reference

#### 5.265.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial\\_heap\\_base\\_/insert\\_fn\\_imps.hpp](#).

### 5.266 `insert_fn_imps.hpp` File Reference

#### 5.266.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin\\_search\\_tree\\_/insert\\_fn\\_imps.hpp](#).

### 5.267 `insert_fn_imps.hpp` File Reference

#### 5.267.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions.

Definition in file [cc\\_hash\\_table\\_map\\_/insert\\_fn\\_imps.hpp](#).

### 5.268 `insert_fn_imps.hpp` File Reference

#### 5.268.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions.

Definition in file [gp\\_hash\\_table\\_map\\_/insert\\_fn\\_imps.hpp](#).

### 5.269 `insert_fn_imps.hpp` File Reference

#### 5.269.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/insert\\_fn\\_imps.hpp](#).

### 5.270 `insert_fn_imps.hpp` File Reference

#### 5.270.1 Detailed Description

Contains implementations of `lu_map_`.

Definition in file [list\\_update\\_map\\_/insert\\_fn\\_imps.hpp](#).

## 5.271 `insert_fn_imps.hpp` File Reference

### 5.271.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov\\_tree\\_map\\_/insert\\_fn\\_imps.hpp](#).

## 5.272 `insert_fn_imps.hpp` File Reference

### 5.272.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing\\_heap\\_/insert\\_fn\\_imps.hpp](#).

## 5.273 `insert_fn_imps.hpp` File Reference

### 5.273.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb\\_tree\\_map\\_/insert\\_fn\\_imps.hpp](#).

## 5.274 `insert_fn_imps.hpp` File Reference

### 5.274.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

Definition in file [rc\\_binomial\\_heap\\_/insert\\_fn\\_imps.hpp](#).

## 5.275 `insert_fn_imps.hpp` File Reference

### 5.275.1 Detailed Description

Contains an implementation class for `splay_tree_`.

Definition in file [splay\\_tree\\_/insert\\_fn\\_imps.hpp](#).

## 5.276 `insert_fn_imps.hpp` File Reference

### 5.276.1 Detailed Description

Contains an implementation for `thin_heap_`.

Definition in file [thin\\_heap\\_/insert\\_fn\\_imps.hpp](#).

## 5.277 `insert_join_fn_imps.hpp` File Reference

### 5.277.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [insert\\_join\\_fn\\_imps.hpp](#).

## 5.278 `insert_no_store_hash_fn_imps.hpp` File Reference

### 5.278.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is not stored.

Definition in file [cc\\_hash\\_table\\_map\\_/insert\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.279 `insert_no_store_hash_fn_imps.hpp` File Reference

### 5.279.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions, when the hash value is not stored.

Definition in file [gp\\_hash\\_table\\_map\\_/insert\\_no\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.280 `insert_store_hash_fn_imps.hpp` File Reference

### 5.280.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is stored.

Definition in file [cc\\_hash\\_table\\_map\\_/insert\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.281 `insert_store_hash_fn_imps.hpp` File Reference

### 5.281.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions, when the hash value is stored.

Definition in file [gp\\_hash\\_table\\_map\\_/insert\\_store\\_hash\\_fn\\_imps.hpp](#).

## 5.282 `iomanip` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_IOMANIP`

## Functions

- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > std::get\_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _-`  
`Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _-`  
`Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_`  
`money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _`  
`CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money<`  
`_MoneyT > __f)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > std::put\_money (const _MoneyT &__mon, bool __intl=false)`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`
- `template<typename _CharT >`  
`_Setfill< _CharT > std::setfill (_CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

## 5.282.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iomanip](#).

## 5.283 ios File Reference

## Macros

- `#define _GLIBCXX_IOS`

## 5.283.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ios](#).

## 5.284 ios\_base.h File Reference

## Classes

- class [std::ios\\_base](#)  
*The base of the I/O class hierarchy.  
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- class [std::ios\\_base::failure](#)  
*These are thrown to indicate problems with io.  
27.4.2.1.1 Class `ios_base::failure`.*

## Namespaces

- namespace [std](#)

## Enumerations

- enum `_ios_Fmtflags` {  
    `_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`,  
    `_S_internal`, `_S_left`, `_S_oct`, `_S_right`,  
    `_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`,  
    `_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`,  
    `_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end` }
- enum `_ios_iostate` {  
    `_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
    `_S_ios_iostate_end` }
- enum `_ios_Openmode` {  
    `_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
    `_S_out`, `_S_trunc`, `_S_ios_openmode_end` }
- enum `_ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }

## Functions

- ios\_base & [std::boolalpha](#) (ios\_base & \_\_base)
- ios\_base & [std::dec](#) (ios\_base & \_\_base)
- ios\_base & [std::fixed](#) (ios\_base & \_\_base)
- ios\_base & [std::hex](#) (ios\_base & \_\_base)
- ios\_base & [std::internal](#) (ios\_base & \_\_base)
- ios\_base & [std::left](#) (ios\_base & \_\_base)
- ios\_base & [std::noboolalpha](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowbase](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowpoint](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowpos](#) (ios\_base & \_\_base)
- ios\_base & [std::noskipws](#) (ios\_base & \_\_base)
- ios\_base & [std::nounitbuf](#) (ios\_base & \_\_base)
- ios\_base & [std::nouppercase](#) (ios\_base & \_\_base)
- ios\_base & [std::oct](#) (ios\_base & \_\_base)
- constexpr \_ios\_Fmtflags [std::operator&](#) (\_ios\_Fmtflags \_\_a, \_ios\_Fmtflags \_\_b)
- constexpr \_ios\_Openmode [std::operator&](#) (\_ios\_Openmode \_\_a, \_ios\_Openmode \_\_b)
- constexpr \_ios\_istate [std::operator&](#) (\_ios\_istate \_\_a, \_ios\_istate \_\_b)
- const \_ios\_Fmtflags & [std::operator&=](#) (\_ios\_Fmtflags & \_\_a, \_ios\_Fmtflags \_\_b)
- const \_ios\_Openmode & [std::operator&=](#) (\_ios\_Openmode & \_\_a, \_ios\_Openmode \_\_b)
- const \_ios\_istate & [std::operator&=](#) (\_ios\_istate & \_\_a, \_ios\_istate \_\_b)
- constexpr \_ios\_Fmtflags [std::operator^](#) (\_ios\_Fmtflags \_\_a, \_ios\_Fmtflags \_\_b)
- constexpr \_ios\_Openmode [std::operator^](#) (\_ios\_Openmode \_\_a, \_ios\_Openmode \_\_b)
- constexpr \_ios\_istate [std::operator^](#) (\_ios\_istate \_\_a, \_ios\_istate \_\_b)
- const \_ios\_Fmtflags & [std::operator^=](#) (\_ios\_Fmtflags & \_\_a, \_ios\_Fmtflags \_\_b)
- const \_ios\_Openmode & [std::operator^=](#) (\_ios\_Openmode & \_\_a, \_ios\_Openmode \_\_b)
- const \_ios\_istate & [std::operator^=](#) (\_ios\_istate & \_\_a, \_ios\_istate \_\_b)
- constexpr \_ios\_Fmtflags [std::operator|](#) (\_ios\_Fmtflags \_\_a, \_ios\_Fmtflags \_\_b)
- constexpr \_ios\_Openmode [std::operator|](#) (\_ios\_Openmode \_\_a, \_ios\_Openmode \_\_b)
- constexpr \_ios\_istate [std::operator|](#) (\_ios\_istate \_\_a, \_ios\_istate \_\_b)
- const \_ios\_Fmtflags & [std::operator|=](#) (\_ios\_Fmtflags & \_\_a, \_ios\_Fmtflags \_\_b)
- const \_ios\_Openmode & [std::operator|=](#) (\_ios\_Openmode & \_\_a, \_ios\_Openmode \_\_b)
- const \_ios\_istate & [std::operator|=](#) (\_ios\_istate & \_\_a, \_ios\_istate \_\_b)
- constexpr \_ios\_Fmtflags [std::operator~](#) (\_ios\_Fmtflags \_\_a)
- constexpr \_ios\_Openmode [std::operator~](#) (\_ios\_Openmode \_\_a)
- constexpr \_ios\_istate [std::operator~](#) (\_ios\_istate \_\_a)
- ios\_base & [std::right](#) (ios\_base & \_\_base)
- ios\_base & [std::scientific](#) (ios\_base & \_\_base)
- ios\_base & [std::showbase](#) (ios\_base & \_\_base)
- ios\_base & [std::showpoint](#) (ios\_base & \_\_base)
- ios\_base & [std::showpos](#) (ios\_base & \_\_base)
- ios\_base & [std::skipws](#) (ios\_base & \_\_base)
- ios\_base & [std::unitbuf](#) (ios\_base & \_\_base)
- ios\_base & [std::uppercase](#) (ios\_base & \_\_base)

## 5.284.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [ios\\_base.h](#).

## 5.285 iosfwd File Reference

## Classes

- class [std::basic\\_filebuf< \\_CharT, \\_Traits >](#)  
*The actual work of input and output (for files).*
- class [std::basic\\_fstream< \\_CharT, \\_Traits >](#)  
*Controlling input and output for files.*
- class [std::basic\\_ifstream< \\_CharT, \\_Traits >](#)  
*Controlling input for files.*
- class [std::basic\\_ofstream< \\_CharT, \\_Traits >](#)  
*Controlling output for files.*

## Namespaces

- namespace [std](#)

## Macros

- `#define \_GLIBCXX\_IOSFWD`

## Typedefs

- typedef [basic\\_filebuf< char >](#) [std::filebuf](#)
- typedef [basic\\_fstream< char >](#) [std::fstream](#)
- typedef [basic\\_ifstream< char >](#) [std::ifstream](#)
- typedef [basic\\_ios< char >](#) [std::ios](#)
- typedef [basic\\_iostream< char >](#) [std::iostream](#)
- typedef [basic\\_istream< char >](#) [std::istream](#)
- typedef [basic\\_istreamstream< char >](#) [std::istreamstream](#)
- typedef [basic\\_ofstream< char >](#) [std::ofstream](#)
- typedef [basic\\_ostream< char >](#) [std::ostream](#)
- typedef [basic\\_ostreamstream< char >](#) [std::ostreamstream](#)
- typedef [basic\\_streambuf< char >](#) [std::streambuf](#)
- typedef [basic\\_stringbuf< char >](#) [std::stringbuf](#)
- typedef [basic\\_stringstream< char >](#) [std::stringstream](#)
- typedef [basic\\_filebuf< wchar\\_t >](#) [std::wfilebuf](#)
- typedef [basic\\_fstream< wchar\\_t >](#) [std::wfstream](#)
- typedef [basic\\_ifstream< wchar\\_t >](#) [std::wifstream](#)
- typedef [basic\\_ios< wchar\\_t >](#) [std::wios](#)
- typedef [basic\\_iostream< wchar\\_t >](#) [std::wiostream](#)
- typedef [basic\\_istream< wchar\\_t >](#) [std::wistream](#)
- typedef [basic\\_istreamstream](#)  
    < [wchar\\_t](#) > [std::wistreamstream](#)
- typedef [basic\\_ofstream< wchar\\_t >](#) [std::wofstream](#)
- typedef [basic\\_ostream< wchar\\_t >](#) [std::wostream](#)
- typedef [basic\\_ostreamstream](#)  
    < [wchar\\_t](#) > [std::wostreamstream](#)
- typedef [basic\\_streambuf< wchar\\_t >](#) [std::wstreambuf](#)
- typedef [basic\\_stringbuf< wchar\\_t >](#) [std::wstringbuf](#)
- typedef [basic\\_stringstream](#)  
    < [wchar\\_t](#) > [std::wstringstream](#)

### 5.285.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iosfwd](#).

## 5.286 **iostream File Reference**

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_IOSTREAM`

### Variables

- static `ios_base::Init` [std::\\_\\_ioinit](#)

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the [HOWTO](#) linked to above.

- `istream` [std::cin](#)
- `ostream` [std::cout](#)
- `ostream` [std::cerr](#)
- `ostream` [std::clog](#)
- `wistream` [std::wcin](#)
- `wostream` [std::wcout](#)
- `wostream` [std::wcerr](#)
- `wostream` [std::wclog](#)

### 5.286.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iostream](#).

## 5.287 **istream File Reference**

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)  
*Template class basic\_istream.*
- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)  
*Template class basic\_istream.*
- class [std::basic\\_istream<\\_CharT, \\_Traits>::sentry](#)  
*Performs setup work for input streams.*

## Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_ISTREAM`

## Functions

- `template<typename _CharT, typename _Traits, typename _Tp>  
basic_istream< _CharT, _Traits> & std::operator>> (basic_istream< _CharT, _Traits> &&__is, _Tp &__x)`
- `template<typename _CharT, typename _Traits>  
basic_istream< _CharT, _Traits> & std::ws (basic_istream< _CharT, _Traits> &__is)`
- `template<typename _CharT, typename _Traits>  
basic_istream< _CharT, _Traits> & std::operator>> (basic_istream< _CharT, _Traits> &__in, _CharT &__c)`
- `template<class _Traits>  
basic_istream< char, _Traits> & std::operator>> (basic_istream< char, _Traits> &__in, unsigned char &__c)`
- `template<class _Traits>  
basic_istream< char, _Traits> & std::operator>> (basic_istream< char, _Traits> &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits>  
basic_istream< _CharT, _Traits> & std::operator>> (basic_istream< _CharT, _Traits> &__in, _CharT *__s)`
- `template<>  
basic_istream< char> & std::operator>> (basic_istream< char> &__in, char *__s)`
- `template<class _Traits>  
basic_istream< char, _Traits> & std::operator>> (basic_istream< char, _Traits> &__in, unsigned char *__s)`
- `template<class _Traits>  
basic_istream< char, _Traits> & std::operator>> (basic_istream< char, _Traits> &__in, signed char *__s)`

## 5.287.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [istream](#).

## 5.288 istream.tcc File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _ISTREAM_TCC`

## Functions

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`

## 5.288.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<istream>`.

Definition in file [istream.tcc](#).

## 5.289 iterator File Reference

## Macros

- `#define \_GLIBCXX\_ITERATOR`

## 5.289.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iterator](#).

## 5.290 iterator File Reference

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define \_EXT\_ITERATOR`

## Functions

- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

### 5.290.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).  
Definition in file [ext/iterator](#).

## 5.291 iterator.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair<\\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple<\\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 5.291.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [iterator.h](#).

## 5.292 iterator.hpp File Reference

### Classes

- class [iterator\\_](#)  
*Range-type iterator.*

### 5.292.1 Detailed Description

Contains an `iterator_` class used for ranging over the elements of the table.

Definition in file [iterator.hpp](#).

## 5.293 iterator\_fn\_imps.hpp File Reference

### 5.293.1 Detailed Description

Contains implementations of `gp_ht_map_`'s iterators related functions, e.g., `begin()`.

Definition in file [iterator\\_fn\\_imps.hpp](#).

## 5.294 iterator\_tracker.h File Reference

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__profile::operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type std::__profile::operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

#### 5.294.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [iterator\\_tracker.h](#).

### 5.295 iterators\_fn\_imps.hpp File Reference

#### 5.295.1 Detailed Description

Contains an implementation class for a binary\_heap.

Definition in file [binary\\_heap\\_/iterators\\_fn\\_imps.hpp](#).

### 5.296 iterators\_fn\_imps.hpp File Reference

#### 5.296.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

Definition in file [bin\\_search\\_tree\\_/iterators\\_fn\\_imps.hpp](#).

### 5.297 iterators\_fn\_imps.hpp File Reference

#### 5.297.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s iterators related functions, e.g., begin().

Definition in file [cc\\_hash\\_table\\_map\\_/iterators\\_fn\\_imps.hpp](#).

### 5.298 iterators\_fn\_imps.hpp File Reference

#### 5.298.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_/iterators\\_fn\\_imps.hpp](#).

### 5.299 iterators\_fn\_imps.hpp File Reference

#### 5.299.1 Detailed Description

Contains implementations of lu\_map\_.

Definition in file [list\\_update\\_map\\_/iterators\\_fn\\_imps.hpp](#).

## 5.300 iterators\_fn\_imps.hpp File Reference

### 5.300.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov\\_tree\\_map\\_/iterators\\_fn\\_imps.hpp](#).

## 5.301 iterators\_fn\_imps.hpp File Reference

### 5.301.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [pat\\_trie\\_/iterators\\_fn\\_imps.hpp](#).

## 5.302 left\_child\_next\_sibling\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >  
*Base class for a basic heap.*

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.302.1 Detailed Description

Contains an implementation class for a basic heap.

Definition in file [left\\_child\\_next\\_sibling\\_heap\\_.hpp](#).

## 5.303 limits File Reference

### Classes

- struct [std::\\_\\_numeric\\_limits\\_base](#)  
*Part of `std::numeric_limits`.*
- struct [std::numeric\\_limits](#)< \_Tp >  
*Properties of fundamental types.*
- struct [std::numeric\\_limits](#)< bool >  
*`numeric_limits<bool>` specialization.*
- struct [std::numeric\\_limits](#)< char >

- numeric\_limits<char> specialization.*
- struct [std::numeric\\_limits< char16\\_t >](#)  
*numeric\_limits<char16\_t> specialization.*
- struct [std::numeric\\_limits< char32\\_t >](#)  
*numeric\_limits<char32\_t> specialization.*
- struct [std::numeric\\_limits< double >](#)  
*numeric\_limits<double> specialization.*
- struct [std::numeric\\_limits< float >](#)  
*numeric\_limits<float> specialization.*
- struct [std::numeric\\_limits< int >](#)  
*numeric\_limits<int> specialization.*
- struct [std::numeric\\_limits< long >](#)  
*numeric\_limits<long> specialization.*
- struct [std::numeric\\_limits< long double >](#)  
*numeric\_limits<long double> specialization.*
- struct [std::numeric\\_limits< long long >](#)  
*numeric\_limits<long long> specialization.*
- struct [std::numeric\\_limits< short >](#)  
*numeric\_limits<short> specialization.*
- struct [std::numeric\\_limits< signed char >](#)  
*numeric\_limits<signed char> specialization.*
- struct [std::numeric\\_limits< unsigned char >](#)  
*numeric\_limits<unsigned char> specialization.*
- struct [std::numeric\\_limits< unsigned int >](#)  
*numeric\_limits<unsigned int> specialization.*
- struct [std::numeric\\_limits< unsigned long >](#)  
*numeric\_limits<unsigned long> specialization.*
- struct [std::numeric\\_limits< unsigned long long >](#)  
*numeric\_limits<unsigned long long> specialization.*
- struct [std::numeric\\_limits< unsigned short >](#)  
*numeric\_limits<unsigned short> specialization.*
- struct [std::numeric\\_limits< wchar\\_t >](#)  
*numeric\_limits<wchar\_t> specialization.*

## Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_digits(T)`
- `#define __glibcxx_digits10(T)`
- `#define __glibcxx_double_has_denorm_loss`
- `#define __glibcxx_double_tinyness_before`
- `#define __glibcxx_double_traps`
- `#define __glibcxx_float_has_denorm_loss`
- `#define __glibcxx_float_tinyness_before`

- `#define __glibcxx_float_traps`
- `#define __glibcxx_integral_traps`
- `#define __glibcxx_long_double_has_denorm_loss`
- `#define __glibcxx_long_double_tinyness_before`
- `#define __glibcxx_long_double_traps`
- `#define __glibcxx_max(T)`
- `#define __glibcxx_max_digits10(T)`
- `#define __glibcxx_min(T)`
- `#define __glibcxx_signed(T)`
- `#define _GLIBCXX_NUMERIC_LIMITS`

#### Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate`, `std::denorm_absent`, `std::denorm_present` }
- enum `std::float_round_style` { `round_indeterminate`, `std::round_toward_zero`, `std::round_to_nearest`, `std::round_toward_infinity`, `std::round_toward_neg_infinity` }

#### 5.303.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [limits](#).

## 5.304 `linear_probe_fn_imp.hpp` File Reference

#### 5.304.1 Detailed Description

Contains a probe policy implementation

Definition in file [linear\\_probe\\_fn\\_imp.hpp](#).

## 5.305 `list` File Reference

#### Macros

- `#define _GLIBCXX_LIST`

#### 5.305.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [list](#).

## 5.306 `list` File Reference

#### Classes

- class `std::__debug::list<_Tp, _Allocator>`  
*Class `std::list` with safety/checking/debug instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_LIST`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

## 5.306.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/list](#).

## 5.307 list File Reference

## Classes

- class [std::\\_\\_profile::list< \\_Tp, \\_Allocator >](#)  
*List wrapper with performance instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Macros

- `#define _GLIBCXX_PROFILE_LIST`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

## 5.307.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/list](#).

## 5.308 list.tcc File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _LIST_TCC`

## 5.308.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

Definition in file [list.tcc](#).

## 5.309 list\_partition.h File Reference

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter >`  
`void \_\_gnu\_parallel::\_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`

- `template<typename _Iter >`  
`void __gnu_parallel::__shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FuncType >`  
`size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FuncType &__f, int __oversampling=0)`

#### 5.309.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [list\\_partition.h](#).

### 5.310 list\_update\_policy.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::lu\\_counter\\_policy< Max\\_Count, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::lu\\_move\\_to\\_front\\_policy< \\_Alloc >](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 5.310.1 Detailed Description

Contains policies for list update containers.

Definition in file [list\\_update\\_policy.hpp](#).

### 5.311 locale File Reference

#### Macros

- `#define _GLIBCXX_LOCALE`

#### 5.311.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [locale](#).

### 5.312 locale\_classes.h File Reference

#### Classes

- class [std::collate< \\_CharT >](#)  
*Facet for localized string comparison.*
- class [std::collate\\_byname< \\_CharT >](#)

*class collate\_byname [22.2.4.2].*

- class [std::locale](#)

*Container class for localization functionality.*

*The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization.*

*A locale is a collection of facets that implement various localization features such as money, time, and number printing.*

- class [std::locale::facet](#)

*Localization functionality base class.*

*The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.*

- class [std::locale::id](#)

*Facet ID class.*

*The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.*

## Namespaces

- namespace [std](#)

### 5.312.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_classes.h](#).

## 5.313 locale\_classes.tcc File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _LOCALE_CLASSES_TCC`

## Functions

- `template<typename _Facet >`  
`bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

### 5.313.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_classes.tcc](#).

## 5.314 locale\_facets.h File Reference

## Classes

- class [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#)  
*Common base for ctype facet.*
- class [std::ctype<\\_CharT>](#)  
*Primary class template ctype facet.  
This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class [std::ctype<char>](#)  
*The ctype<char> specialization.  
This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.*
- class [std::ctype<wchar\\_t>](#)  
*The ctype<wchar\_t> specialization.  
This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.*
- class [std::ctype\\_byname<\\_CharT>](#)  
*class ctype\_byname [22.2.1.2].*
- class [std::ctype\\_byname<char>](#)  
*22.2.1.4 Class ctype\_byname specializations.*
- class [std::num\\_get<\\_CharT, \\_InIter>](#)  
*Primary class template num\_get.  
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class [std::num\\_put<\\_CharT, \\_OutIter>](#)  
*Primary class template num\_put.  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- class [std::numpunct<\\_CharT>](#)  
*Primary class template numpunct.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class [std::numpunct\\_byname<\\_CharT>](#)  
*class numpunct\_byname [22.2.3.2].*

## Namespaces

- namespace [std](#)

## Macros

- `#define \_GLIBCXX\_NUM\_FACETS`

## Functions

- `template<typename _CharT>  
\_CharT \* std::add\_grouping (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`

- `template<typename _Tp >`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > std::__write (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _Outlter >`  
`_Outlter std::__write (_Outlter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`

#### 5.314.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets.h](#).

## 5.315 locale\_facets.tcc File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _LOCALE_FACETS_TCC`

## Functions

- `template<typename _CharT >`  
`_CharT * std::add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT`  
`* __first, const _CharT * __last)`
- `template<typename _CharT, typename _ValueT >`  
`int std::int_to_char (_CharT * __bufend, _ValueT __v, const _CharT * __lit, ios_base::fmtflags __flags, bool`  
`__dec)`
- `bool std::verify_grouping (const char * __grouping, size_t __grouping_size, const string & __grouping_tmp)`  
`throw ()`

## 5.315.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets.tcc](#).

## 5.316 locale\_facets\_nonio.h File Reference

## Classes

- class [std::messages< \\_CharT >](#)  
*Primary class template messages.*  
*This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [std::messages\\_base](#)  
*Messages facet base class providing catalog typedef.*
- class [std::messages\\_byname< \\_CharT >](#)  
*class messages\_byname [22.2.7.2].*
- class [std::money\\_base](#)  
*Money format ordering data.*  
*This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*
- class [std::money\\_get< \\_CharT, \\_InIter >](#)  
*Primary class template money\_get.*  
*This facet encapsulates the code to parse and return a monetary amount from a string.*
- class [std::money\\_put< \\_CharT, \\_OutIter >](#)  
*Primary class template money\_put.*  
*This facet encapsulates the code to format and output a monetary amount.*
- class [std::moneypunct< \\_CharT, \\_Intl >](#)  
*Primary class template moneypunct.*  
*This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*
- class [std::moneypunct\\_byname< \\_CharT, \\_Intl >](#)  
*class moneypunct\_byname [22.2.6.4].*

- class [std::time\\_base](#)  
*Time format ordering data.*  
*This class provides an enum representing different orderings of time: day, month, and year.*
- class [std::time\\_get<\\_CharT, \\_InIter >](#)  
*Primary class template time\_get.*  
*This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*
- class [std::time\\_get\\_byname<\\_CharT, \\_InIter >](#)  
*class time\_get\_byname [22.2.5.2].*
- class [std::time\\_put<\\_CharT, \\_OutIter >](#)  
*Primary class template time\_put.*  
*This facet encapsulates the code to format and output dates and times according to formats used by strftime().*
- class [std::time\\_put\\_byname<\\_CharT, \\_OutIter >](#)  
*class time\_put\_byname [22.2.5.4].*

#### Namespaces

- namespace [std](#)

#### 5.316.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets\\_nonio.h](#).

### 5.317 locale\_facets\_nonio.tcc File Reference

#### Namespaces

- namespace [std](#)

#### Macros

- `#define _LOCALE_FACETS_NONIO_TCC`

#### 5.317.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets\\_nonio.tcc](#).

### 5.318 localefwd.h File Reference

#### Classes

- class [std::codecvt<\\_InternT, \\_ExternT, \\_StateT >](#)

Primary class template codecvt.

NB: Generic, mostly useless implementation.

- class `std::codecvt_byname<_InternT, _ExternT, _StateT>`

class codecvt\_byname [22.2.1.6].

- class `std::collate<_CharT>`

Facet for localized string comparison.

- class `std::collate_byname<_CharT>`

class collate\_byname [22.2.4.2].

- class `std::ctype<_CharT>`

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

- class `std::ctype_byname<_CharT>`

class ctype\_byname [22.2.1.2].

- class `std::messages<_CharT>`

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

- class `std::messages_byname<_CharT>`

class messages\_byname [22.2.7.2].

- class `std::money_get<_CharT, _InIter>`

Primary class template money\_get.

This facet encapsulates the code to parse and return a monetary amount from a string.

- class `std::money_put<_CharT, _OutIter>`

Primary class template money\_put.

This facet encapsulates the code to format and output a monetary amount.

- class `std::moneypunct<_CharT, _Intl>`

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class `std::moneypunct_byname<_CharT, _Intl>`

class moneypunct\_byname [22.2.6.4].

- class `std::num_get<_CharT, _InIter>`

Primary class template num\_get.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

- class `std::num_put<_CharT, _OutIter>`

Primary class template num\_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

- class `std::numpunct<_CharT>`

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

- class `std::numpunct_byname<_CharT>`

class numpunct\_byname [22.2.3.2].

- class `std::time_get<_CharT, _InIter>`

Primary class template time\_get.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class `std::time_get_byname<_CharT, _InIter>`

*class time\_get\_byname [22.2.5.2].*

- class `std::time_put< _CharT, _Outlter >`

*Primary class template time\_put.*

*This facet encapsulates the code to format and output dates and times according to formats used by strftime().*

- class `std::time_put_byname< _CharT, _Outlter >`

*class time\_put\_byname [22.2.5.4].*

## Namespaces

- namespace `std`

## Functions

- template<typename \_Facet >  
bool `std::has_facet` (const locale &\_\_loc) throw ()
- template<typename \_CharT >  
bool `std::isalnum` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isalpha` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::iscntrl` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isdigit` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isgraph` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::islower` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isprint` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::ispunct` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isspace` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isupper` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
bool `std::isxdigit` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
\_CharT `std::tolower` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT >  
\_CharT `std::toupper` (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_Facet >  
const \_Facet & `std::use_facet` (const locale &\_\_loc)

### 5.318.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file `localefwd.h`.

## 5.319 losertree.h File Reference

## Classes

- class [\\_\\_gnu\\_parallel::\\_LoserTree< \\_\\_stable, \\_Tp, \\_Compare >](#)  
*Stable \_LoserTree variant.*
- class [\\_\\_gnu\\_parallel::\\_LoserTree< false, \\_Tp, \\_Compare >](#)  
*Unstable \_LoserTree variant.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >](#)  
*Guarded loser/tournament tree.*
- struct [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_Loser](#)  
*Internal representation of a \_LoserTree element.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >](#)  
*Stable \_LoserTree implementation.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)  
*Unstable \_LoserTree implementation.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >](#)  
*Base class of \_Loser Tree implementation using pointers.*
- struct [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_Loser](#)  
*Internal representation of \_LoserTree \_\_elements.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)  
*Stable unguarded \_LoserTree variant storing pointers.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)  
*Unstable unguarded \_LoserTree variant storing pointers.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >](#)  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)  
*Stable implementation of unguarded \_LoserTree.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)  
*Non-Stable implementation of unguarded \_LoserTree.*
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >](#)  
*Base class for unguarded \_LoserTree implementation.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## 5.319.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [losertree.h](#).

## 5.320 lu\_counter\_metadata.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >](#)  
*A list-update metadata type that moves elements to the front of the list based on the counter algorithm.*
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)  
*Base class for list-update counter policy.*

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 5.320.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

Definition in file [lu\\_counter\\_metadata.hpp](#).

## 5.321 lu\_map.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >](#)  
*list-based (with updates) associative container. Skip to the lu, my darling.*

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

#### 5.321.1 Detailed Description

Contains a list update map.

Definition in file [lu\\_map.hpp](#).

## 5.322 macros.h File Reference

## Macros

- #define `__glibcxx_check_bucket_index`(\_N)
- #define `__glibcxx_check_equal_allocs`(\_Other)
- #define `__glibcxx_check_erase`(\_Position)
- #define `__glibcxx_check_erase_after`(\_Position)
- #define `__glibcxx_check_erase_range`(\_First, \_Last)
- #define `__glibcxx_check_erase_range_after`(\_First, \_Last)
- #define `__glibcxx_check_heap`(\_First, \_Last)
- #define `__glibcxx_check_heap_pred`(\_First, \_Last, \_Pred)
- #define `__glibcxx_check_insert`(\_Position)
- #define `__glibcxx_check_insert_after`(\_Position)
- #define `__glibcxx_check_insert_range`(\_Position, \_First, \_Last)
- #define `__glibcxx_check_insert_range_after`(\_Position, \_First, \_Last)
- #define `__glibcxx_check_max_load_factor`(\_F)
- #define `__glibcxx_check_non_empty_range`(\_First, \_Last)
- #define `__glibcxx_check_nonempty`()
- #define `__glibcxx_check_partitioned_lower`(\_First, \_Last, \_Value)
- #define `__glibcxx_check_partitioned_lower_pred`(\_First, \_Last, \_Value, \_Pred)
- #define `__glibcxx_check_partitioned_upper`(\_First, \_Last, \_Value)
- #define `__glibcxx_check_partitioned_upper_pred`(\_First, \_Last, \_Value, \_Pred)
- #define `__glibcxx_check_self_move_assign`(\_Other)
- #define `__glibcxx_check_sorted`(\_First, \_Last)
- #define `__glibcxx_check_sorted_pred`(\_First, \_Last, \_Pred)
- #define `__glibcxx_check_sorted_set`(\_First1, \_Last1, \_First2)
- #define `__glibcxx_check_sorted_set_pred`(\_First1, \_Last1, \_First2, \_Pred)
- #define `__glibcxx_check_string`(\_String)
- #define `__glibcxx_check_string_len`(\_String, \_Len)
- #define `__glibcxx_check_subscript`(\_N)
- #define `__glibcxx_check_valid_range`(\_First, \_Last)
- #define `__GLIBCXX_DEBUG_VERIFY`(\_Condition, \_ErrorMessage)
- #define `__GLIBCXX_DEBUG_VERIFY_AT`(\_Condition, \_ErrorMessage, \_File, \_Line)

## 5.322.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [macros.h](#).

## 5.322.2 Macro Definition Documentation

5.322.2.1 #define `__glibcxx_check_erase`( *\_Position* )

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 135 of file `macros.h`.

#### 5.322.2.2 #define \_\_glibcxx\_check\_erase\_after( *\_Position* )

Verify that we can erase the element after the iterator *\_Position*. We can erase the element if the *\_Position* iterator is before a dereferenceable one and references this sequence.

Definition at line 149 of file macros.h.

#### 5.322.2.3 #define \_\_glibcxx\_check\_erase\_range( *\_First*, *\_Last* )

Verify that we can erase the elements in the iterator range [*\_First*, *\_Last*). We can erase the elements if [*\_First*, *\_Last*) is a valid iterator range within this sequence.

Definition at line 163 of file macros.h.

#### 5.322.2.4 #define \_\_glibcxx\_check\_erase\_range\_after( *\_First*, *\_Last* )

Verify that we can erase the elements in the iterator range (*\_First*, *\_Last*). We can erase the elements if (*\_First*, *\_Last*) is a valid iterator range within this sequence.

Definition at line 175 of file macros.h.

#### 5.322.2.5 #define \_\_glibcxx\_check\_heap\_pred( *\_First*, *\_Last*, *\_Pred* )

Verify that the iterator range [*\_First*, *\_Last*) is a heap w.r.t. the predicate *\_Pred*.

Definition at line 314 of file macros.h.

#### 5.322.2.6 #define \_\_glibcxx\_check\_insert( *\_Position* )

Verify that we can insert into *\*this* with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 73 of file macros.h.

#### 5.322.2.7 #define \_\_glibcxx\_check\_insert\_after( *\_Position* )

Verify that we can insert into *\*this* after the iterator *\_Position*. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 90 of file macros.h.

#### 5.322.2.8 #define \_\_glibcxx\_check\_insert\_range( *\_Position*, *\_First*, *\_Last* )

Verify that we can insert the values in the iterator range [*\_First*, *\_Last*) into *\*this* with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [*\_First*, *\_Last*) is a valid (possibly empty) range. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

**Todo** We would like to be able to check for noninterference of *\_Position* and the range [*\_First*, *\_Last*), but that can't (in general) be done.

Definition at line 110 of file macros.h.

5.322.2.9 `#define __glibcxx_check_insert_range_after( _Position, _First, _Last )`

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

**Todo** We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Definition at line 127 of file `macros.h`.

5.322.2.10 `#define __glibcxx_check_partitioned_lower( _First, _Last, _Value )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

Definition at line 262 of file `macros.h`.

5.322.2.11 `#define __glibcxx_check_partitioned_lower_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 282 of file `macros.h`.

5.322.2.12 `#define __glibcxx_check_partitioned_upper_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 294 of file `macros.h`.

5.322.2.13 `#define __glibcxx_check_sorted_pred( _First, _Last, _Pred )`

Verify that the iterator range `[_First, _Last)` is sorted by the predicate `_Pred`.

Definition at line 233 of file `macros.h`.

5.322.2.14 `#define _GLIBCXX_DEBUG_VERIFY_AT( _Condition, _ErrorMessage, _File, _Line )`

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 41 of file `macros.h`.

## 5.323 malloc\_allocator.h File Reference

## Classes

- class `__gnu_cxx::malloc_allocator< _Tp >`  
*An allocator that uses malloc.*  
*This is precisely the allocator defined in the C++ Standard.*

## Namespaces

- namespace `__gnu_cxx`

## Functions

- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

### 5.323.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [malloc\\_allocator.h](#).

## 5.324 map File Reference

### Macros

- `#define _GLIBCXX_MAP`

### 5.324.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [map](#).

## 5.325 map File Reference

### Macros

- `#define _GLIBCXX_DEBUG_MAP`

### 5.325.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map](#).

## 5.326 map File Reference

### Macros

- `#define _GLIBCXX_PROFILE_MAP`

### 5.326.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map](#).

## 5.327 map.h File Reference

### Classes

- class [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class std::map with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
bool **std::\_\_debug::operator!=** (const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
bool **std::\_\_debug::operator<** (const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
bool **std::\_\_debug::operator<=** (const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
bool **std::\_\_debug::operator==** (const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
bool **std::\_\_debug::operator>** (const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
bool **std::\_\_debug::operator>=** (const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key , typename \_Tp , typename \_Compare , typename \_Allocator >  
void **std::\_\_debug::swap** (map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)

#### 5.327.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map.h](#).

## 5.328 map.h File Reference

### Classes

- class [std::\\_\\_profile::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class std::map wrapper with performance instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
void std::__profile::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

## 5.328.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map.h](#).

## 5.329 mask\_array.h File Reference

## Classes

- class [std::mask\\_array< \\_Tp >](#)  
*Reference to selected subset of an array.*

## Namespaces

- namespace [std](#)

## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 5.329.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [mask\\_array.h](#).

## 5.330 mask\_based\_range\_hashing.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >](#)  
*Range hashing policy.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.330.1 Detailed Description

Contains a range hashing policy base.

Definition in file [mask\\_based\\_range\\_hashing.hpp](#).

## 5.331 memory File Reference

## Macros

- `#define _GLIBCXX_MEMORY`

## 5.331.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [memory](#).

## 5.332 memory File Reference

## Classes

- struct [\\_\\_gnu\\_cxx::temporary\\_buffer< \\_ForwardIterator, \\_Tp >](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_MEMORY`

## Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _InputIter __first, _Size __count, _-`  
`ForwardIter __result, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`  
`pair< _RandomAccessIter,`  
`_ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _RandomAccessIter __first, _Size __count, _ForwardIter`  
`__result, std::random\_access\_iterator\_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _InputIter __first, _Size __count, _-`  
`ForwardIter __result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _InputIter __first, _Size __count, _-`  
`ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _InputIter __first, _Size __count, _-`  
`ForwardIter __result, std::allocator< _Tp >)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::uninitialized\_copy\_n ( _InputIter __first, _Size __count, _ForwardIter`  
`__result)`

## 5.332.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/memory](#).

## 5.333 merge.h File Reference

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_gnu\_parallel::merge\_advance ( _RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2,`  
`_RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_gnu\_parallel::merge\_advance\_movc ( _RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__-`  
`begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_gnu\_parallel::merge\_advance\_usual ( _RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__-`  
`begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::parallel\_merge\_advance ( _RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__-`  
`begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max-`  
`length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::parallel\_merge\_advance ( _RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__-`  
`begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max-`  
`length, _Compare __comp)`

## 5.333.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [merge.h](#).

5.334 `messages_members.h` File Reference

## Namespaces

- namespace [std](#)

## 5.334.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [messages\\_members.h](#).

5.335 `mod_based_range_hashing.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing<Size\\_Type>](#)  
*Mod based range hashing.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## 5.335.1 Detailed Description

Contains a range hashing policy base.

Definition in file [mod\\_based\\_range\\_hashing.hpp](#).

5.336 `move.h` File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_FORWARD(_Tp, __val)`
- `#define _GLIBCXX_MOVE(__val)`

## Functions

- `template<typename _Tp >`  
`_Tp * std::\_\_addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`  
`_Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr`  
`std::remove\_reference< _Tp >`  
`::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr conditional`  
`< __move_if_noexcept_cond< _Tp >`  
`::value, const _Tp &, _Tp && >`  
`::type std::move\_if\_noexcept (_Tp &__x) noexcept`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >`
- `template<typename _Tp, size_t _Nm>`  
`void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a`

## Variables

- **`std::__a`**
- `void`  
`is_nothrow_move_assignable`  
`< _Tp >::value _Tp std::__tmp`

## 5.336.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Definition in file [move.h](#).

## 5.337 mt\_allocator.h File Reference

## Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_common\\_pool\\_policy](#)< \_PoolTp, \_Thread >  
*Policy for shared \_\_pool objects.*
- class [\\_\\_gnu\\_cxx::\\_\\_mt\\_alloc](#)< \_Tp, \_Poolp >  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- class [\\_\\_gnu\\_cxx::\\_\\_mt\\_alloc\\_base](#)< \_Tp >  
*Base class for \_Tp dependent member functions.*
- struct [\\_\\_gnu\\_cxx::\\_\\_per\\_type\\_pool\\_policy](#)< \_Tp, \_PoolTp, \_Thread >

*Policy for individual `__pool` objects.*

- class `__gnu_cxx::__pool< false >`

*Specialization for single thread.*

- class `__gnu_cxx::__pool< true >`

*Specialization for thread enabled, via `gthreads.h`.*

- struct `__gnu_cxx::__pool_base`

*Base class for pool object.*

- class `__pool< _Thread >`

*Data describing the underlying memory pool, parameterized on threading support.*

## Namespaces

- namespace `__gnu_cxx`

## Macros

- `#define __thread_default`

## Typedefs

- typedef `void(* __gnu_cxx::__destroy_handler )(void *)`

## Functions

- template<typename `_Tp` , typename `_Poolp` >  
`bool __gnu_cxx::operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- template<typename `_Tp` , typename `_Poolp` >  
`bool __gnu_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`

### 5.337.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [mt\\_allocator.h](#).

## 5.338 `multimap.h` File Reference

### Classes

- class `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`  
*Class `std::multimap` with safety/checking/debug instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__debug`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

## 5.338.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multimap.h](#).

## 5.339 multimap.h File Reference

## Classes

- class [std::\\_\\_profile::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class std::multimap wrapper with performance instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap<`  
`_Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap<`  
`_Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap<`  
`_Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap<`  
`_Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__profile::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _-`  
`Compare, _Allocator > &__rhs)`

### 5.339.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multimap.h](#).

## 5.340 multiseq\_selection.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_Lexicographic< \\_T1, \\_T2, \\_Compare >](#)  
*Compare \_\_a pair of types lexicographically, ascending.*
- class [\\_\\_gnu\\_parallel::\\_LexicographicReverse< \\_T1, \\_T2, \\_Compare >](#)  
*Compare \_\_a pair of types lexicographically, descending.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define __S(__i)`
- `#define __S(__i)`

### Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`  
`void \_\_gnu\_parallel::multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`  
`_RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::-`  
`iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp \_\_gnu\_parallel::multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`  
`_RankType &__offset, _Compare __comp=std::less< _Tp >())`

## 5.340.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets. The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiseq\\_selection.h](#).

## 5.341 multiset.h File Reference

## Classes

- class [std::\\_\\_debug::multiset<\\_Key, \\_Compare, \\_Allocator>](#)  
*Class std::multiset with safety/checking/debug instrumentation.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

## 5.341.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multiset.h](#).

## 5.342 multiset.h File Reference

### Classes

- class [std::\\_\\_profile::multiset<\\_Key, \\_Compare, \\_Allocator>](#)

*Class std::multiset wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

#### 5.342.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multiset.h](#).

## 5.343 multiway\_merge.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch<\\_\\_sentinels, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare>](#)  
*Switch for 3-way merging with \_\_sentinels turned off.*
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare>](#)

*Switch for 3-way merging with \_\_sentinels turned on.*

- struct `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 4-way merging with \_\_sentinels turned off.*

- struct `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 4-way merging with \_\_sentinels turned on.*

- struct `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for k-way merging with \_\_sentinels turned on.*

- struct `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterlIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for k-way merging with \_\_sentinels turned off.*

- class `__gnu_parallel::_GuardedlIterator< _RAIter, _Compare >`

*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*

- struct `__gnu_parallel::_LoserTreeTraits< _Tp >`

*Traits for determining whether the loser tree should use pointers or copies.*

- struct `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

*Stable sorting functor.*

- struct `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

*Non-\_\_stable sorting functor.*

## Namespaces

- namespace `__gnu_parallel`

## Macros

- `#define _GLIBCXX_PARALLEL_DECISION(__a, __b, __c, __d)`
- `#define _GLIBCXX_PARALLEL_LENGTH(__s)`
- `#define _GLIBCXX_PARALLEL_MERGE_3_CASE(__a, __b, __c, __c0, __c1)`
- `#define _GLIBCXX_PARALLEL_MERGE_4_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputlIterator, typename _DifferenceTp, typename _Compare > _OutputlIterator __gnu_parallel::__merge_advance(_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputlIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<bool __stable, bool __sentinels, typename _RAIterlIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::__sequential_multiway_merge(_RAIterlIterator __seqs_begin, _RAIterlIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterlIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairlIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut __gnu_parallel::multiway_merge(_RAIterPairlIterator __seqs_begin, _RAIterPairlIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairlIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut __gnu_parallel::multiway_merge(_RAIterPairlIterator __seqs_begin, _RAIterPairlIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void \_\_gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void \_\_gnu\_parallel::multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`

- ```
template<typename __stable, typename __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter,
typename _Compare >
_RAlter3 gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,
_RAlter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_
threads)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __
seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __
seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::exact_tag __tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __
seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __
seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __
seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter-
PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel-
::sequential_tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter-
PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel-
::exact_tag __tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter-
PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter-
PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __
tag=parallel_tag(0))

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
_RAlterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter-
PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag
__tag)
```

5.343.1 Detailed Description

Implementation of sequential and parallel multiway merge. Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. *ACM Journal of Experimental Algorithmics*, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway_merge.h](#).

5.343.2 Macro Definition Documentation

5.343.2.1 `#define GLIBCXX_PARALLEL_LENGTH(__s)`

Length of a sequence described by a pair of iterators.

Definition at line 54 of file `multiway_merge.h`.

Referenced by `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

5.344 multiway_mergesort.h File Reference

Classes

- struct [__gnu_parallel::Piece<_DifferenceTp>](#)
Subsequence description.
- struct [__gnu_parallel::PMWMSortingData<_RAIter>](#)
Data accessed by all threads.
- struct [__gnu_parallel::SplitConsistently<__exact, _RAIter, _Compare, _SortingPlacesIterator>](#)
Split consistently.
- struct [__gnu_parallel::SplitConsistently<false, _RAIter, _Compare, _SortingPlacesIterator>](#)
Split by sampling.
- struct [__gnu_parallel::SplitConsistently<true, _RAIter, _Compare, _SortingPlacesIterator>](#)
Split by exact splitting.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- template<typename _RAIter, typename _DifferenceTp>
void [__gnu_parallel::__determine_samples](#) (_PMWMSortingData<_RAIter> * __sd, _DifferenceTp __num_samples)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void [__gnu_parallel::parallel_sort_mwms](#) (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void [__gnu_parallel::parallel_sort_mwms_pu](#) (_PMWMSortingData<_RAIter> * __sd, _Compare & __comp)

5.344.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway_mergesort.h](#).

5.345 mutex File Reference

Classes

- struct [std::adopt_lock_t](#)

Assume the calling thread has already obtained mutex ownership and manage it.

- struct `std::defer_lock_t`

Do not acquire ownership of the mutex.

- class `std::lock_guard< _Mutex >`

Scoped lock idiom.

- class `std::mutex`

mutex

- struct `std::once_flag`

once_flag

- class `std::recursive_mutex`

recursive_mutex

- struct `std::try_to_lock_t`

Try to acquire ownership of the mutex without blocking.

- class `std::unique_lock< _Mutex >`

unique_lock

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_MUTEX`

Functions

- `mutex & std::__get_once_mutex ()`
- `void std::__once_proxy (void)`
- `void std::__set_once_functor_lock_ptr (unique_lock< mutex > *)`
- `template<typename _Lock > unique_lock< _Lock > std::__try_to_lock (_Lock &__l)`
- `template<typename _Callable, typename... _Args> void std::call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)`
- `template<typename _L1, typename _L2, typename... _L3> void std::lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)`
- `template<typename _Mutex > void std::swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3> int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`

Variables

- `function< void()> std::__once_functor`
- `constexpr adopt_lock_t std::adopt_lock`
- `constexpr defer_lock_t std::defer_lock`
- `constexpr try_to_lock_t std::try_to_lock`

5.345.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [mutex](#).

5.346 `nested_exception.h` File Reference

Classes

- class [std::nested_exception](#)
Exception class with `exception_ptr` data member.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Ex >`
`const nested_exception * std::__get_nested_exception (const _Ex &__ex)`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&, const nested_exception *e) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&...) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void std::rethrow_if_nested (const _Ex &__ex)`
- `void std::rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`
`void std::throw_with_nested (_Ex __ex)`

5.346.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

Definition in file [nested_exception.h](#).

5.347 `new` File Reference

Classes

- class [std::bad_alloc](#)
*Exception possibly thrown by `new`.
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*

Namespaces

- namespace [std](#)

Typedefs

- typedef void(* [std::new_handler](#))()

Functions

- new_handler [std::set_new_handler](#) (new_handler) throw ()
- void * [operator new](#) (std::size_t) __attribute__((__externally_visible__))
- void * [operator new\[\]](#) (std::size_t) __attribute__((__externally_visible__))
- void [operator delete](#) (void *) noexcept __attribute__((__externally_visible__))
- void [operator delete\[\]](#) (void *) noexcept __attribute__((__externally_visible__))
- void * [operator new](#) (std::size_t, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void * [operator new\[\]](#) (std::size_t, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void [operator delete](#) (void *, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void [operator delete\[\]](#) (void *, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void * [operator new](#) (std::size_t, void *__p) noexcept
- void * [operator new\[\]](#) (std::size_t, void *__p) noexcept
- void [operator delete](#) (void *, void *) noexcept
- void [operator delete\[\]](#) (void *, void *) noexcept

Variables

- const nothrow_t **std::nothrow**

5.347.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4 for more.

Definition in file [new](#).

5.347.2 Function Documentation

5.347.2.1 void operator delete (void *) [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.2 void operator delete (void *, const std::nothrow_t &) [nothrow]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.3 void operator delete (void *, void *) [inline], [nothrow]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 115 of file new.

5.347.2.4 void operator delete[] (void *) [nothrow]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.5 void operator delete[] (void *, const std::nothrow_t &) [nothrow]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.6 `void operator delete[] (void *, void *) [inline], [nothrow]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 116 of file `new`.

5.347.2.7 `void* operator new (std::size_t)`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.8 `void* operator new (std::size_t, const std::nothrow_t &) [nothrow]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.9 `void* operator new (std::size_t, void * __p) [inline], [nothrow]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 109 of file `new`.

5.347.2.10 void* operator new[](std::size_t)

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.11 void* operator new[](std::size_t, const std::nothrow_t &) [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.12 void* operator new[](std::size_t, void * __p) [inline], [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 111 of file `new`.

5.348 new_allocator.h File Reference

Classes

- class [__gnu_cxx::new_allocator< _Tp >](#)
*An allocator that uses global new, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`

5.348.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [new_allocator.h](#).

5.349 node.hpp File Reference

Classes

- `struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >`
Node.

Namespaces

- namespace [__gnu_pbds](#)

5.349.1 Detailed Description

Contains an implementation struct for this type of heap's node.

Definition in file [left_child_next_sibling_heap_/node.hpp](#).

5.350 node.hpp File Reference

Classes

- `struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >`
Node for Red-Black trees.

Namespaces

- namespace [__gnu_pbds](#)

5.350.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb_tree_map_/node.hpp](#).

5.351 node.hpp File Reference

Classes

- struct [__gnu_pbds::detail::splay_tree_node_](#) < Value_Type, Metadata, _Alloc >
Node for splay tree.

Namespaces

- namespace [__gnu_pbds](#)

5.351.1 Detailed Description

Contains an implementation struct for `splay_tree_`'s node.

Definition in file [splay_tree_/node.hpp](#).

5.352 node_iterators.hpp File Reference

Classes

- class [__gnu_pbds::detail::bin_search_tree_const_node_it_](#) < Node, Const_Iterator, Iterator, _Alloc >
Const node iterator.
- class [__gnu_pbds::detail::bin_search_tree_node_it_](#) < Node, Const_Iterator, Iterator, _Alloc >
Node iterator.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC`
- `#define PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC`

5.352.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin_search_tree_/node_iterators.hpp](#).

5.353 node_iterators.hpp File Reference

Classes

- class [__gnu_pbds::detail::ov_tree_node_const_it_](#) < Value_Type, Metadata_Type, _Alloc >
Const node reference.
- class [__gnu_pbds::detail::ov_tree_node_it_](#) < Value_Type, Metadata_Type, _Alloc >
Node reference.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC`
- `#define PB_DS_OV_TREE_NODE_ITERATOR_C_DEC`

5.353.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov_tree_map_/node_iterators.hpp](#).

5.354 node_metadata_selector.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_metadata_helper](#)< Node_Update, false >
Specialization, false.
- struct [__gnu_pbds::detail::tree_metadata_helper](#)< Node_Update, true >
Specialization, true.
- struct [__gnu_pbds::detail::tree_node_metadata_dispatch](#)< Key, Data, Cmp_Fn, Node_Update, _Alloc >
Tree node metadata dispatch.
- struct [tree_metadata_helper](#)< Node_Update, _BTp >
Tree metadata helper.

Namespaces

- namespace [__gnu_pbds](#)

5.354.1 Detailed Description

Contains an implementation class for trees.

Definition in file [tree_policy/node_metadata_selector.hpp](#).

5.355 node_metadata_selector.hpp File Reference

Classes

- struct [__gnu_pbds::detail::trie_metadata_helper](#)< Node_Update, false >
Specialization, false.
- struct [__gnu_pbds::detail::trie_metadata_helper](#)< Node_Update, true >
Specialization, true.
- struct [__gnu_pbds::detail::trie_node_metadata_dispatch](#)< Key, Data, Cmp_Fn, Node_Update, _Alloc >
Trie node metadata dispatch.
- struct [trie_metadata_helper](#)< Node_Update, _BTp >
Trie metadata helper.

Namespaces

- namespace [__gnu_pbds](#)

5.355.1 Detailed Description

Contains an implementation class for tries.

Definition in file [trie_policy/node_metadata_selector.hpp](#).

5.356 null_node_metadata.hpp File Reference

Classes

- struct [__gnu_pbds::detail::dumnode_const_iterator](#)< Key, Data, _Alloc >
Constant node iterator.

Namespaces

- namespace [__gnu_pbds](#)

5.356.1 Detailed Description

Contains an implementation class for tree-like classes.

Definition in file [null_node_metadata.hpp](#).

5.357 numeric File Reference

Macros

- `#define _GLIBCXX_NUMERIC`

5.357.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [numeric](#).

5.358 numeric File Reference

Namespaces

- namespace [__gnu_cxx](#)

Macros

- `#define _EXT_NUMERIC`

Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp gnu_cxx::power (_Tp __x, _Integer __n)`

5.358.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

Definition in file [ext/numeric](#).

5.359 numeric File Reference

Namespaces

- namespace [std](#)
- namespace [std::parallel](#)

Macros

- `#define _GLIBCXX_PARALLEL_NUMERIC_H`

Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Tp std::parallel::accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`
`_Tp std::parallel::accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >`
`_Tp std::parallel::accumulate_switch (_RAIter __begin, _RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::parallel::adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::parallel::adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::parallel::inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_unbalanced)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Tp std::__parallel::inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::__parallel::partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 > _Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator > _OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator > _OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`

5.359.1 Detailed Description

Parallel STL function calls corresponding to `stl_numeric.h`. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/numeric](#).

5.360 `numeric_traits.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

Macros

- `#define __glibcxx_digits(_Tp)`
- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max(_Tp)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __glibcxx_min(_Tp)`
- `#define __glibcxx_signed(_Tp)`

5.360.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [numeric_traits.h](#).

5.361 `numeric_fwd.h` File Reference

Namespaces

- namespace `std`
- namespace `std::__parallel`

Functions

- `template<typename _Iter, typename _Tp, typename _Tag >`
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >`
`_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`

- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`

5.361.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [numericfwd.h](#).

5.362 omp_loop.h File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`
`_Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op -`
`__o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::`
`difference_type __bound)`

5.362.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp_loop.h](#).

5.363 `omp_loop_static.h` File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end,
_Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

5.363.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp_loop_static.h](#).

5.364 `opt_random.h` File Reference

Namespaces

- namespace [std](#)

5.364.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

Definition in file [bits/opt_random.h](#).

5.365 `order_statistics_imp.hpp` File Reference

5.365.1 Detailed Description

Contains forward declarations for `order_statistics_key`

Definition in file [tree_policy/order_statistics_imp.hpp](#).

5.366 `order_statistics_imp.hpp` File Reference

5.366.1 Detailed Description

Contains forward declarations for `order_statistics_key`

Definition in file [trie_policy/order_statistics_imp.hpp](#).

5.367 os_defines.h File Reference

Macros

- #define `__NO_CTYPE`

5.367.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [os_defines.h](#).

5.368 ostream File Reference

Classes

- class [std::basic_ostream<_CharT, _Traits>](#)
Template class basic_ostream.
- class [std::basic_ostream<_CharT, _Traits>::sentry](#)
Performs setup work for output streams.

Namespaces

- namespace [std](#)

Macros

- #define `_GLIBCXX_OSTREAM`

Functions

- `template<typename _CharT, typename _Traits>`
`basic_ostream<_CharT, _Traits> & std::endl (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits>`
`basic_ostream<_CharT, _Traits> & std::ends (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits>`
`basic_ostream<_CharT, _Traits> & std::flush (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits, typename _Tp>`
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits>`
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__out, char __c)`
- `template<class _Traits>`
`basic_ostream<char, _Traits> & std::operator<< (basic_ostream<char, _Traits> &__out, char __c)`
- `template<class _Traits>`
`basic_ostream<char, _Traits> & std::operator<< (basic_ostream<char, _Traits> &__out, signed char __c)`

- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

5.368.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ostream](#).

5.369 ostream.tcc File Reference

Namespaces

- namespace [std](#)

Macros

- `#define _OSTREAM_TCC`

Functions

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`

5.369.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

Definition in file [ostream.tcc](#).

5.370 ostream_insert.h File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT * __s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`
`void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT * __s, streamsize __n)`

5.370.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

Definition in file [ostream_insert.h](#).

5.371 ov_tree_map.hpp File Reference

Classes

- class [__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >](#)
Ordered-vector tree associative-container.
- class [__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >](#)
Conditional destructor.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_NODE_ITERATOR_NAME`
- `#define PB_DS_OV_TREE_NAME`
- `#define PB_DS_OV_TREE_TRAITS_BASE`

5.371.1 Detailed Description

Contains an implementation class for `ov_tree`.

Definition in file [ov_tree_map.hpp](#).

5.372 pairing_heap_.hpp File Reference

Classes

- class [__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- namespace [__gnu_pbds](#)

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_P_HEAP_BASE**

5.372.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing_heap_.hpp](#).

5.373 par_loop.h File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >
[__Op __gnu_parallel::__for_each_template_random_access_ed](#) (_RAIter __begin, _RAIter __end, _Op __o, _Fu
&__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type
__bound)

5.373.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [par_loop.h](#).

5.374 parallel.h File Reference

5.374.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel.h](#).

5.375 `partial_sum.h` File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::difference_type __n)`

5.375.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partial_sum.h](#).

5.376 `partition.h` File Reference

Namespaces

- namespace [__gnu_parallel](#)

Macros

- `#define _GLIBCXX_VOLATILE`

Functions

- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Predicate >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`

5.376.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partition.h](#).

5.376.2 Macro Definition Documentation

5.376.2.1 `#define _GLIBCXX_VOLATILE`

Decide whether to declare certain variables volatile.

Definition at line 43 of file [partition.h](#).

Referenced by `__gnu_parallel::__parallel_partition()`.

5.377 pat_trie_.hpp File Reference

Classes

- class [__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >](#)
PATRICIA trie.
This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_NODE_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_PAT_TRIE_NAME`
- `#define PB_DS_PAT_TRIE_TRAITS_BASE`
- `#define PB_DS_RECURSIVE_COUNT_LEAFS(X)`

5.377.1 Detailed Description

Contains an implementation class for a patricia tree.

Definition in file [pat_trie_.hpp](#).

5.378 pat_trie_base.hpp File Reference

Classes

- struct [__gnu_pbds::detail::pat_trie_base](#)
Base type for PATRICIA trees.

- class [__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >](#)
Const iterator.
- struct [__gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >](#)
Head node for PATRICIA tree.
- struct [__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >](#)
Internal node type, PATRICIA tree.
- struct [__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator](#)
Constant child iterator.
- struct [__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator](#)
Child iterator.
- class [__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >](#)
Iterator.
- struct [__gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >](#)
Leaf node for PATRICIA tree.
- struct [__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >](#)
Metadata base primary template.
- struct [__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >](#)
Specialization for null metadata.
- struct [__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >](#)
Node base.
- class [__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >](#)
Node const iterator.
- class [__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >](#)
Node iterator.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- **#define PB_DS_CLASS_C_DEC**
- **#define PB_DS_CLASS_T_DEC**
- **#define PB_DS_CONST_IT_C_DEC**
- **#define PB_DS_CONST_ODIR_IT_C_DEC**
- **#define PB_DS_IT_C_DEC**
- **#define PB_DS_ODIR_IT_C_DEC**
- **#define PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC**
- **#define PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC**

5.378.1 Detailed Description

Contains the base class for a patricia tree.

Definition in file [pat_trie_base.hpp](#).

5.379 pod_char_traits.h File Reference

Classes

- struct [__gnu_cxx::character< V, I, S >](#)
A POD class that serves as a character abstraction class.
- struct [std::char_traits< __gnu_cxx::character< V, I, S > >](#)
char_traits<__gnu_cxx::character> specialization.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- template<typename V, typename I, typename S >
bool [__gnu_cxx::operator<](#) (const character< V, I, S > &lhs, const character< V, I, S > &rhs)
- template<typename V, typename I, typename S >
bool [__gnu_cxx::operator==](#) (const character< V, I, S > &lhs, const character< V, I, S > &rhs)

5.379.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pod_char_traits.h](#).

5.380 point_const_iterator.hpp File Reference

Classes

- class [__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >](#)
Const point-type iterator.

Namespaces

- namespace [__gnu_pbds](#)

5.380.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file [binary_heap_/point_const_iterator.hpp](#).

5.381 point_const_iterator.hpp File Reference

Classes

- class [__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >](#)
Const point-type iterator.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.381.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file [left_child_next_sibling_heap/point_const_iterator.hpp](#).

5.382 `point_const_iterator.hpp` File Reference

Classes

- class [point_const_iterator_](#)
Const point-type iterator.

5.382.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

Definition in file [unordered_iterator/point_const_iterator.hpp](#).

5.383 `point_iterator.hpp` File Reference

Classes

- class [point_iterator_](#)
Find type iterator.

5.383.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

Definition in file [point_iterator.hpp](#).

5.384 `point_iterators.hpp` File Reference

Classes

- class [__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#)
Const iterator.

- class [__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#)
Iterator.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

5.384.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [point_iterators.hpp](#).

5.385 `pointer.h` File Reference

Classes

- struct [__gnu_cxx::_Invalid_type](#)
- class [__gnu_cxx::_Pointer_adapter< _Storage_policy >](#)
- class [__gnu_cxx::_Relative_pointer_impl< _Tp >](#)
A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.
- class [__gnu_cxx::_Relative_pointer_impl< const _Tp >](#)
- class [__gnu_cxx::_Std_pointer_impl< _Tp >](#)
A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.
- struct [__gnu_cxx::_Unqualified_type< _Tp >](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Macros

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

Functions

- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`

- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

5.385.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

Definition in file [pointer.h](#).

5.386 policy_access_fn_imps.hpp File Reference

5.386.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary_heap_/policy_access_fn_imps.hpp](#).

5.387 policy_access_fn_imps.hpp File Reference

5.387.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin_search_tree_/policy_access_fn_imps.hpp](#).

5.388 policy_access_fn_imps.hpp File Reference

5.388.1 Detailed Description

Contains implementations of `cc_ht_map_`'s policy access functions.

Definition in file [cc_hash_table_map_/policy_access_fn_imps.hpp](#).

5.389 `policy_access_fn_imps.hpp` File Reference

5.389.1 Detailed Description

Contains implementations of `gp_ht_map_`'s policy access functions.

Definition in file [gp_hash_table_map_/policy_access_fn_imps.hpp](#).

5.390 `policy_access_fn_imps.hpp` File Reference

5.390.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left_child_next_sibling_heap_/policy_access_fn_imps.hpp](#).

5.391 `policy_access_fn_imps.hpp` File Reference

5.391.1 Detailed Description

Contains an implementation class for `ov_tree`.

Definition in file [ov_tree_map_/policy_access_fn_imps.hpp](#).

5.392 `policy_access_fn_imps.hpp` File Reference

5.392.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [pat_trie_/policy_access_fn_imps.hpp](#).

5.393 `pool_allocator.h` File Reference

Classes

- class [__gnu_cxx::__pool_alloc<_Tp>](#)
Allocator using a memory pool with a single lock.
- class [__gnu_cxx::__pool_alloc_base](#)
Base class for `__pool_alloc`.

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Tp>`
`bool __gnu_cxx::operator!= (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`
- `template<typename _Tp>`
`bool __gnu_cxx::operator== (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`

5.393.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pool_allocator.h](#).

5.394 `postypes.h` File Reference

Classes

- class [std::fpos<_StateT>](#)
Class representing stream positions.

Namespaces

- namespace [std](#)

Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate_t > [std::streampos](#)
- typedef ptrdiff_t [std::streamsize](#)
- typedef fpos< mbstate_t > [std::u16streampos](#)
- typedef fpos< mbstate_t > [std::u32streampos](#)
- typedef fpos< mbstate_t > [std::wstreampos](#)

Functions

- template<typename _StateT>
bool **std::operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _StateT>
bool [std::operator==](#) (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)

5.394.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [postypes.h](#).

5.395 `prefix_search_node_update_imp.hpp` File Reference

5.395.1 Detailed Description

Contains an implementation of `prefix_search_node_update`.

Definition in file [prefix_search_node_update_imp.hpp](#).

5.396 `priority_queue.hpp` File Reference

Classes

- class [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>](#)

Namespaces

- namespace [__gnu_pbds](#)

5.396.1 Detailed Description

Contains `priority_queues`.

Definition in file [priority_queue.hpp](#).

5.397 `priority_queue_base_dispatch.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>](#)
Specialization for binary_heap.
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>](#)
Specialization for binomial_heap.
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>](#)
Specialization for pairing_heap.
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type>](#)
>
Specialization for rc_binomial_heap.
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>](#)
Specialization for thin_heap.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`

5.397.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file [priority_queue_base_dispatch.hpp](#).

5.398 `probe_fn_base.hpp` File Reference

Classes

- class [__gnu_pbds::detail::probe_fn_base<_Alloc>](#)
Probe functor base.

Namespaces

- namespace [__gnu_pbds](#)

5.398.1 Detailed Description

Contains a probe policy base.

Definition in file [probe_fn_base.hpp](#).

5.399 `profiler.h` File Reference

Classes

- struct [__gnu_profile::__reentrance_guard](#)
Reentrance guard.

Namespaces

- namespace [__gnu_profile](#)

Macros

- `#define __profcxx_hashtable_construct(__x...)`
- `#define __profcxx_hashtable_construct2(__x...)`
- `#define __profcxx_hashtable_destruct(__x...)`
- `#define __profcxx_hashtable_destruct2(__x...)`
- `#define __profcxx_hashtable_resize(__x...)`
- `#define __profcxx_is_invalid()`
- `#define __profcxx_is_off()`
- `#define __profcxx_is_on()`
- `#define __profcxx_list_construct(__x...)`
- `#define __profcxx_list_construct2(__x...)`
- `#define __profcxx_list_destruct(__x...)`
- `#define __profcxx_list_destruct2(__x...)`
- `#define __profcxx_list_insert(__x...)`
- `#define __profcxx_list_invalid_operator(__x...)`
- `#define __profcxx_list_iterate(__x...)`
- `#define __profcxx_list_operation(__x...)`
- `#define __profcxx_list_rewind(__x...)`
- `#define __profcxx_map_to_unordered_map_construct(__x...)`
- `#define __profcxx_map_to_unordered_map_destruct(__x...)`

- `#define __profcxx_map_to_unordered_map_erase(__x...)`
- `#define __profcxx_map_to_unordered_map_find(__x...)`
- `#define __profcxx_map_to_unordered_map_insert(__x...)`
- `#define __profcxx_map_to_unordered_map_invalidate(__x...)`
- `#define __profcxx_map_to_unordered_map_iterate(__x...)`
- `#define __profcxx_report()`
- `#define __profcxx_turn_off()`
- `#define __profcxx_turn_on()`
- `#define __profcxx_vector_construct(__x...)`
- `#define __profcxx_vector_construct2(__x...)`
- `#define __profcxx_vector_destruct(__x...)`
- `#define __profcxx_vector_destruct2(__x...)`
- `#define __profcxx_vector_find(__x...)`
- `#define __profcxx_vector_insert(__x...)`
- `#define __profcxx_vector_invalid_operator(__x...)`
- `#define __profcxx_vector_iterate(__x...)`
- `#define __profcxx_vector_resize(__x...)`
- `#define __profcxx_vector_resize2(__x...)`
- `#define _GLIBCXX_PROFILE_DATA(__name)`
- `#define _GLIBCXX_PROFILE_DEFINE_DATA(__type, __name, __initial_value...)`
- `#define _GLIBCXX_PROFILE_DEFINE_UNINIT_DATA(__type, __name)`
- `#define _GLIBCXX_PROFILE_MAX_STACK_DEPTH`
- `#define _GLIBCXX_PROFILE_MAX_STACK_DEPTH_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT`
- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR`
- `#define _GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__trace_hash_func_construct (const void *)`
- `void __gnu_profile::__trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_construct (const void *, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_construct (const void *)`
- `void __gnu_profile::__trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_set_find (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_set_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_destruct (const void *)`

- void `__gnu_profile::__trace_list_to_slist_operation` (const void *)
- void `__gnu_profile::__trace_list_to_slist_rewind` (const void *)
- void `__gnu_profile::__trace_list_to_vector_construct` (const void *)
- void `__gnu_profile::__trace_list_to_vector_destruct` (const void *)
- void `__gnu_profile::__trace_list_to_vector_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_list_to_vector_invalid_operator` (const void *)
- void `__gnu_profile::__trace_list_to_vector_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_list_to_vector_resize` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_construct` (const void *)
- void `__gnu_profile::__trace_map_to_unordered_map_destruct` (const void *)
- void `__gnu_profile::__trace_map_to_unordered_map_erase` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_find` (const void *, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void *)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_size_construct` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_size_resize` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_construct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_find` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void *)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void *, std::size_t, std::size_t)
- bool `__gnu_profile::__turn_off` ()
- bool `__gnu_profile::__turn_on` ()

5.399.1 Detailed Description

Interface of the profiling runtime library.

Definition in file [profiler.h](#).

5.400 profiler_algos.h File Reference

Namespaces

- namespace [__gnu_profile](#)

Functions

- template<typename _InputIterator, typename _Function >
_Function `__gnu_profile::__for_each` (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _Container >
void `__gnu_profile::__insert_top_n` (_Container &__output, const typename _Container::value_type &__value, typename _Container::size_type __n)
- template<typename _ForwardIterator, typename _Tp >
_ForwardIterator `__gnu_profile::__remove` (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- `template<typename _Container >`
`void __gnu_profile::__top_n (const _Container &__input, _Container &__output, typename _Container::size_type __n)`

5.400.1 Detailed Description

Algorithms used by the profile extension. This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

Definition in file [profiler_algos.h](#).

5.401 profiler_container_size.h File Reference

Classes

- class [__gnu_profile::__container_size_info](#)
A container size instrumentation line in the object table.
- class [__gnu_profile::__container_size_stack_info](#)
A container size instrumentation line in the stack table.
- class [__gnu_profile::__trace_container_size](#)
Container size instrumentation trace producer.

Namespaces

- namespace [__gnu_profile](#)

5.401.1 Detailed Description

Diagnostics for container sizes.

Definition in file [profiler_container_size.h](#).

5.402 profiler_hash_func.h File Reference

Classes

- class [__gnu_profile::__hashfunc_info](#)
A hash performance instrumentation line in the object table.
- class [__gnu_profile::__hashfunc_stack_info](#)
A hash performance instrumentation line in the stack table.
- class [__gnu_profile::__trace_hash_func](#)
Hash performance instrumentation producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void `__gnu_profile::__trace_hash_func_construct` (const void *)
- void `__gnu_profile::__trace_hash_func_destruct` (const void *, std::size_t, std::size_t, std::size_t)
- void `__gnu_profile::__trace_hash_func_init` ()
- void `__gnu_profile::__trace_hash_func_report` (FILE *__f, __warning_vector_t &__warnings)

5.402.1 Detailed Description

Data structures to represent profiling traces.

Definition in file [profiler_hash_func.h](#).

5.403 profiler_hashtable_size.h File Reference

Classes

- class [__gnu_profile::__trace_hashtable_size](#)
Hashtable size instrumentation trace producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void `__gnu_profile::__trace_hashtable_size_construct` (const void *, std::size_t)
- void `__gnu_profile::__trace_hashtable_size_destruct` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_hashtable_size_init` ()
- void `__gnu_profile::__trace_hashtable_size_report` (FILE *__f, __warning_vector_t &__warnings)
- void `__gnu_profile::__trace_hashtable_size_resize` (const void *, std::size_t, std::size_t)

5.403.1 Detailed Description

Collection of hashtable size traces.

Definition in file [profiler_hashtable_size.h](#).

5.404 profiler_list_to_slist.h File Reference

Namespaces

- namespace [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_list_to_slist_construct](#) (const void *)
- void [__gnu_profile::__trace_list_to_slist_destruct](#) (const void *)
- void [__gnu_profile::__trace_list_to_slist_init](#) ()
- void [__gnu_profile::__trace_list_to_slist_operation](#) (const void *)
- void [__gnu_profile::__trace_list_to_slist_report](#) (FILE * __f, __warning_vector_t & __warnings)
- void [__gnu_profile::__trace_list_to_slist_rewind](#) (const void *)

5.404.1 Detailed Description

Diagnostics for list to slist.

Definition in file [profiler_list_to_slist.h](#).

5.405 profiler_list_to_vector.h File Reference

Classes

- class [__gnu_profile::__list2vector_info](#)
A list-to-vector instrumentation line in the object table.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_list_to_vector_construct](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_destruct](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_init](#) ()
- void [__gnu_profile::__trace_list_to_vector_insert](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_list_to_vector_invalid_operator](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_iterate](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_list_to_vector_report](#) (FILE * __f, __warning_vector_t & __warnings)
- void [__gnu_profile::__trace_list_to_vector_resize](#) (const void *, std::size_t, std::size_t)

5.405.1 Detailed Description

diagnostics for list to vector.

Definition in file [profiler_list_to_vector.h](#).

5.406 profiler_map_to_unordered_map.h File Reference

Classes

- class [__gnu_profile::__map2umap_info](#)
A map-to-unordered_map instrumentation line in the object table.

- class [__gnu_profile::__map2umap_stack_info](#)
A map-to-unordered_map instrumentation line in the stack table.
- class [__gnu_profile::__trace_map2umap](#)
Map-to-unordered_map instrumentation producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- int [__gnu_profile::__log2](#) (std::size_t __size)
- float [__gnu_profile::__map_erase_cost](#) (std::size_t __size)
- float [__gnu_profile::__map_find_cost](#) (std::size_t __size)
- float [__gnu_profile::__map_insert_cost](#) (std::size_t __size)
- void [__gnu_profile::__trace_map_to_unordered_map_construct](#) (const void *)
- void [__gnu_profile::__trace_map_to_unordered_map_destruct](#) (const void *)
- void [__gnu_profile::__trace_map_to_unordered_map_erase](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_find](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_init](#) ()
- void [__gnu_profile::__trace_map_to_unordered_map_insert](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_invalidate](#) (const void *)
- void [__gnu_profile::__trace_map_to_unordered_map_iterate](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_map_to_unordered_map_report](#) (FILE *__f, __warning_vector_t &__warnings)

5.406.1 Detailed Description

Diagnostics for map to unordered_map.

Definition in file [profiler_map_to_unordered_map.h](#).

5.407 profiler_node.h File Reference

Classes

- class [__gnu_profile::__object_info_base](#)
Base class for a line in the object table.
- class [__gnu_profile::__stack_hash](#)
Hash function for summary trace using call stack as index.
- class [__gnu_profile::__stack_info_base](#)< [__object_info](#) >
Base class for a line in the stack table.

Namespaces

- namespace [__gnu_profile](#)

Typedefs

- typedef void * **__gnu_profile::__instruction_address_t**
- typedef const void * **__gnu_profile::__object_t**
- typedef std::vector< **__instruction_address_t** > **__gnu_profile::__stack_npt**
- typedef **__stack_npt** * **__gnu_profile::__stack_t**

Functions

- **__stack_t** **__gnu_profile::__get_stack** ()
- std::size_t **__gnu_profile::__size** (**__stack_t** __stack)
- std::size_t **__gnu_profile::__stack_max_depth** ()
- void **__gnu_profile::__write** (FILE * __f, **__stack_t** __stack)

5.407.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file [profiler_node.h](#).

5.408 profiler_state.h File Reference

Namespaces

- namespace [__gnu_profile](#)

Enumerations

- enum **__state_type** { **__ON**, **__OFF**, **__INVALID** }

Functions

- bool **__gnu_profile::__is_invalid** ()
- bool **__gnu_profile::__is_off** ()
- bool **__gnu_profile::__is_on** ()
- bool **__gnu_profile::__turn** (**__state_type** __s)
- bool **__gnu_profile::__turn_off** ()
- bool **__gnu_profile::__turn_on** ()
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (**__state_type**, **__state**, **__INVALID**)

5.408.1 Detailed Description

Global profiler state.

Definition in file [profiler_state.h](#).

5.409 profiler_trace.h File Reference

Classes

- class [__gnu_profile::__trace_base](#)< [__object_info](#), [__stack_info](#) >
Base class for all trace producers.
- struct [__gnu_profile::__warning_data](#)
Representation of a warning.

Namespaces

- namespace [__gnu_profile](#)

Macros

- #define [_GLIBCXX_IMPL_UNORDERED_MAP](#)

Typedefs

- typedef std::vector
 < [__cost_factor](#) * > [__gnu_profile::__cost_factor_vector](#)
- typedef std::unordered_map
 < [std::string](#), [std::string](#) > [__gnu_profile::__env_t](#)
- typedef std::vector
 < [__warning_data](#) > [__gnu_profile::__warning_vector_t](#)

Functions

- [std::size_t __gnu_profile::__env_to_size_t](#) (const char * __env_var, [std::size_t](#) __default_value)
- [int __gnu_profile::__log_magnitude](#) (float __f)
- [std::size_t __gnu_profile::__max_mem](#) ()
- [FILE * __gnu_profile::__open_output_file](#) (const char * __extension)
- [bool __gnu_profile::__profcxx_init](#) ()
- [void __gnu_profile::__profcxx_init_unconditional](#) ()
- [void __gnu_profile::__read_cost_factors](#) ()
- [void __gnu_profile::__report](#) (void)
- [void __gnu_profile::__set_cost_factors](#) ()
- [void __gnu_profile::__set_max_mem](#) ()
- [void __gnu_profile::__set_max_stack_trace_depth](#) ()
- [void __gnu_profile::__set_max_warn_count](#) ()
- [void __gnu_profile::__set_trace_path](#) ()
- [std::size_t __gnu_profile::__stack_max_depth](#) ()
- [void __gnu_profile::__trace_hash_func_init](#) ()
- [void __gnu_profile::__trace_hash_func_report](#) (FILE * __f, [__warning_vector_t](#) & __warnings)
- [void __gnu_profile::__trace_hashtable_size_init](#) ()
- [void __gnu_profile::__trace_hashtable_size_report](#) (FILE * __f, [__warning_vector_t](#) & __warnings)
- [void __gnu_profile::__trace_list_to_slist_init](#) ()
- [void __gnu_profile::__trace_list_to_slist_report](#) (FILE * __f, [__warning_vector_t](#) & __warnings)
- [void __gnu_profile::__trace_list_to_vector_init](#) ()

- void `__gnu_profile::__trace_list_to_vector_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__gnu_profile::__trace_map_to_unordered_map_init` ()
- void `__gnu_profile::__trace_map_to_unordered_map_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__gnu_profile::__trace_vector_size_init` ()
- void `__gnu_profile::__trace_vector_size_report` (FILE *, __warning_vector_t &)
- void `__gnu_profile::__trace_vector_to_list_init` ()
- void `__gnu_profile::__trace_vector_to_list_report` (FILE *, __warning_vector_t &)
- void `__gnu_profile::__write_cost_factors` ()
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_hash_func *, __S_hash_func, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_hashtable_size *, __S_hashtable_size, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_map2umap *, __S_map2umap, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_vector_size *, __S_vector_size, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_vector_to_list *, __S_vector_to_list, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_list_to_slist *, __S_list_to_slist, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__trace_list_to_vector *, __S_list_to_vector, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __vector_shift_cost_factor, {"__vector_shift_cost_factor", 1.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __vector_iterate_cost_factor, {"__vector_iterate_cost_factor", 1.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __vector_resize_cost_factor, {"__vector_resize_cost_factor", 1.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __list_shift_cost_factor, {"__list_shift_cost_factor", 0.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __list_iterate_cost_factor, {"__list_iterate_cost_factor", 10.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __list_resize_cost_factor, {"__list_resize_cost_factor", 0.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __map_insert_cost_factor, {"__map_insert_cost_factor", 1.5})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __map_erase_cost_factor, {"__map_erase_cost_factor", 1.5})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __map_find_cost_factor, {"__map_find_cost_factor", 1})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __map_iterate_cost_factor, {"__map_iterate_cost_factor", 2.3})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __umap_insert_cost_factor, {"__umap_insert_cost_factor", 12.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __umap_erase_cost_factor, {"__umap_erase_cost_factor", 12.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __umap_find_cost_factor, {"__umap_find_cost_factor", 10.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor, __umap_iterate_cost_factor, {"__umap_iterate_cost_factor", 1.7})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (__cost_factor_vector *, __cost_factors, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (const char *, __S_trace_file_name, __GLIBCXX_PROFILE_TRACE_PATH_ROOT)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (std::size_t, __S_max_warn_count, __GLIBCXX_PROFILE_MAX_WARN_COUNT)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (std::size_t, __S_max_stack_depth, __GLIBCXX_PROFILE_MAX_STACK_DEPTH)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (std::size_t, __S_max_mem, __GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (__env_t, __env)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (__gnu_cxx::__mutex, __global_lock)

5.409.1 Detailed Description

Data structures to represent profiling traces.

Definition in file [profiler_trace.h](#).

5.410 profiler_vector_size.h File Reference

Classes

- class [__gnu_profile::__trace_vector_size](#)
Hashtable size instrumentation trace producer.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_vector_size_construct](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_vector_size_destruct](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_vector_size_init](#) ()
- void [__gnu_profile::__trace_vector_size_report](#) (FILE *, __warning_vector_t &)
- void [__gnu_profile::__trace_vector_size_resize](#) (const void *, std::size_t, std::size_t)

5.410.1 Detailed Description

Collection of vector size traces.

Definition in file [profiler_vector_size.h](#).

5.411 profiler_vector_to_list.h File Reference

Classes

- class [__gnu_profile::__trace_vector_to_list](#)
Vector-to-list instrumentation producer.
- class [__gnu_profile::__vector2list_info](#)
A vector-to-list instrumentation line in the object table.
- class [__gnu_profile::__vector2list_stack_info](#)
A vector-to-list instrumentation line in the stack table.

Namespaces

- namespace [__gnu_profile](#)

Functions

- void `__gnu_profile::__trace_vector_to_list_construct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_find` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_init` ()
- void `__gnu_profile::__trace_vector_to_list_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void *)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_report` (FILE *, __warning_vector_t &)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void *, std::size_t, std::size_t)

5.411.1 Detailed Description

diagnostics for vector to list.

Definition in file [profiler_vector_to_list.h](#).

5.412 ptr_traits.h File Reference

Classes

- struct [std::pointer_traits<_Ptr>](#)
Uniform interface to all pointer-like types.
- struct [std::pointer_traits<_Tp*>](#)
Partial specialization for built-in pointers.

Namespaces

- namespace [std](#)

5.412.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [ptr_traits.h](#).

5.413 quadratic_probe_fn_imp.hpp File Reference

5.413.1 Detailed Description

Contains a probe policy implementation

Definition in file [quadratic_probe_fn_imp.hpp](#).

5.414 queue File Reference

Macros

- `#define _GLIBCXX_QUEUE`

5.414.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [queue](#).

5.415 queue.h File Reference

Classes

- class [__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >](#)
Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Namespaces

- namespace [__gnu_parallel](#)

Macros

- `#define` [_GLIBCXX_VOLATILE](#)

5.415.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [queue.h](#).

5.415.2 Macro Definition Documentation

5.415.2.1 `#define` _GLIBCXX_VOLATILE

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file [queue.h](#).

5.416 quicksort.h File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Compare >`
`void` [__gnu_parallel::__parallel_sort_qs](#) (`_RAIter` __begin, `_RAIter` __end, `_Compare` __comp, `_ThreadIndex` __num_threads)
- `template<typename _RAIter, typename _Compare >`
`void` [__gnu_parallel::__parallel_sort_qs_conquer](#) (`_RAIter` __begin, `_RAIter` __end, `_Compare` __comp, `_ThreadIndex` __num_threads)

- `template<typename _RAIter, typename _Compare >`
`std::iterator_traits< _RAIter >`
`::difference_type __gnu_parallel::__parallel_qs_divide` (`_RAIter __begin`, `_RAIter __end`, `_Compare __-`
`comp`, `typename std::iterator_traits< _RAIter >::difference_type __pivot_rank`, `typename std::iterator_traits< -`
`_RAIter >::difference_type __num_samples`, `_ThreadIndex __num_threads`)

5.416.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [quicksort.h](#).

5.417 `r_erase_fn_imps.hpp` File Reference

5.417.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

Definition in file [bin_search_tree_/r_erase_fn_imps.hpp](#).

5.418 `r_erase_fn_imps.hpp` File Reference

5.418.1 Detailed Description

Contains an implementation class for `pat_trie`.

Definition in file [pat_trie_/r_erase_fn_imps.hpp](#).

5.419 `random` File Reference

Macros

- `#define _GLIBCXX_RANDOM`

5.419.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [random](#).

5.420 `random.h` File Reference

Classes

- class [std::bernoulli_distribution](#)
A Bernoulli random number distribution.
- struct [std::bernoulli_distribution::param_type](#)
- class [std::binomial_distribution< _IntType >](#)
A discrete binomial random number distribution.

- struct `std::binomial_distribution<_IntType>::param_type`
- class `std::cauchy_distribution<_RealType>`
A cauchy_distribution random number distribution.
- struct `std::cauchy_distribution<_RealType>::param_type`
- class `std::chi_squared_distribution<_RealType>`
A chi_squared_distribution random number distribution.
- struct `std::chi_squared_distribution<_RealType>::param_type`
- class `std::discard_block_engine<_RandomNumberEngine, __p, __r>`
- class `std::discrete_distribution<_IntType>`
A discrete_distribution random number distribution.
- struct `std::discrete_distribution<_IntType>::param_type`
- class `std::exponential_distribution<_RealType>`
An exponential continuous distribution for random numbers.
- struct `std::exponential_distribution<_RealType>::param_type`
- class `std::extreme_value_distribution<_RealType>`
A extreme_value_distribution random number distribution.
- struct `std::extreme_value_distribution<_RealType>::param_type`
- class `std::fisher_f_distribution<_RealType>`
A fisher_f_distribution random number distribution.
- struct `std::fisher_f_distribution<_RealType>::param_type`
- class `std::gamma_distribution<_RealType>`
A gamma continuous distribution for random numbers.
- struct `std::gamma_distribution<_RealType>::param_type`
- class `std::geometric_distribution<_IntType>`
A discrete geometric random number distribution.
- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`
A model of a linear congruential random number generator.
- class `std::lognormal_distribution<_RealType>`
A lognormal_distribution random number distribution.
- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>`
- class `std::negative_binomial_distribution<_IntType>`
A negative_binomial_distribution random number distribution.
- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`
A normal continuous distribution for random numbers.
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`
A piecewise_constant_distribution random number distribution.
- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`
A piecewise_linear_distribution random number distribution.
- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`
A discrete Poisson random number distribution.

- struct [std::poisson_distribution<_IntType>::param_type](#)
- class [std::random_device](#)
- class [std::seed_seq](#)
The seed_seq class generates sequences of seeds for random number generators.
- class [std::shuffle_order_engine<_RandomNumberEngine, __k>](#)
Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.
- class [std::student_t_distribution<_RealType>](#)
A student_t_distribution random number distribution.
- struct [std::student_t_distribution<_RealType>::param_type](#)
- class [std::uniform_int_distribution<_IntType>](#)
Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.
- struct [std::uniform_int_distribution<_IntType>::param_type](#)
- class [std::uniform_real_distribution<_RealType>](#)
Uniform continuous distribution for random numbers.
- struct [std::uniform_real_distribution<_RealType>::param_type](#)
- class [std::weibull_distribution<_RealType>](#)
A weibull_distribution random number distribution.
- struct [std::weibull_distribution<_RealType>::param_type](#)

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine
 < minstd_rand0, 256 > **std::knuth_b**
- typedef
 linear_congruential_engine
 < uint_fast32_t, 48271UL, 0UL, 2147483647UL > [std::minstd_rand](#)
- typedef
 linear_congruential_engine
 < uint_fast32_t, 16807UL, 0UL, 2147483647UL > [std::minstd_rand0](#)
- typedef
 mersenne_twister_engine
 < uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef
 mersenne_twister_engine
 < uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000-ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [std::mt19937_64](#)
- typedef discard_block_engine
 < ranlux24_base, 223, 23 > **std::ranlux24**
- typedef
 subtract_with_carry_engine
 < uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**

- typedef discard_block_engine
< ranlux48_base, 389, 11 > **std::ranlux48**
- typedef
subtract_with_carry_engine
< uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >
_RealType **std::generate_canonical** (_UniformRandomNumberGenerator &__g)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool **std::operator!=** (const **std::linear_congruential_engine**< _UIntType, __a, __c, __m > &__lhs, const **std::linear_congruential_engine**< _UIntType, __a, __c, __m > &__rhs)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool **std::operator!=** (const **std::mersenne_twister_engine**< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const **std::mersenne_twister_engine**< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool **std::operator!=** (const **std::subtract_with_carry_engine**< _UIntType, __w, __s, __r > &__lhs, const **std::subtract_with_carry_engine**< _UIntType, __w, __s, __r > &__rhs)
- template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool **std::operator!=** (const **std::discard_block_engine**< _RandomNumberEngine, __p, __r > &__lhs, const **std::discard_block_engine**< _RandomNumberEngine, __p, __r > &__rhs)
- template<typename _RandomNumberEngine, size_t __w, typename _UIntType >
bool **std::operator!=** (const **std::independent_bits_engine**< _RandomNumberEngine, __w, _UIntType > &__lhs, const **std::independent_bits_engine**< _RandomNumberEngine, __w, _UIntType > &__rhs)
- template<typename _RandomNumberEngine, size_t __k>
bool **std::operator!=** (const **std::shuffle_order_engine**< _RandomNumberEngine, __k > &__lhs, const **std::shuffle_order_engine**< _RandomNumberEngine, __k > &__rhs)
- template<typename _IntType >
bool **std::operator!=** (const **std::uniform_int_distribution**< _IntType > &__d1, const **std::uniform_int_distribution**< _IntType > &__d2)
- template<typename _IntType >
bool **std::operator!=** (const **std::uniform_real_distribution**< _IntType > &__d1, const **std::uniform_real_distribution**< _IntType > &__d2)
- template<typename _RealType >
bool **std::operator!=** (const **std::normal_distribution**< _RealType > &__d1, const **std::normal_distribution**< _RealType > &__d2)
- template<typename _RealType >
bool **std::operator!=** (const **std::lognormal_distribution**< _RealType > &__d1, const **std::lognormal_distribution**< _RealType > &__d2)
- template<typename _RealType >
bool **std::operator!=** (const **std::gamma_distribution**< _RealType > &__d1, const **std::gamma_distribution**< _RealType > &__d2)
- template<typename _RealType >
bool **std::operator!=** (const **std::chi_squared_distribution**< _RealType > &__d1, const **std::chi_squared_distribution**< _RealType > &__d2)
- template<typename _RealType >
bool **std::operator!=** (const **std::cauchy_distribution**< _RealType > &__d1, const **std::cauchy_distribution**< _RealType > &__d2)

- `template<typename _RealType >`
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)`

- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &_`
`_x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution<`
`_IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution<`
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _`
`RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_`
`distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _Real`
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType`
`> &__x)`
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _Real`
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _`
`RealType > &__x)`

5.420.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

Definition in file [random.h](#).

5.421 random.tcc File Reference

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Macros

- `#define _RANDOM_TCC`

Functions

- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >
_OutputIterator std::__detail::__normalize (_InputIterator __first, _InputIterator __last, _OutputIterator __result,
const _Tp &__factor)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >
_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_
engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UInt-
Type __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine<
_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_
engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _
RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _
RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_
distribution< _IntType > &__x)`

- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _Int-`
`Type > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _Int-`
`Type > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _-`
`IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _Real-`
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution<`
`_RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution<`
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution<`
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _-`
`RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _-`
`RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _-`
`RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _Int-`
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_-`
`distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution<`
`_RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_-`
`distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &_-`
`__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution<`
`_IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution<`
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _-`
`RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_-`
`distribution< _RealType > &__x)`
- `template<typename _RealType >`
`bool std::operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _-`
`RealType > &__d2)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _U-`
`IntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UInt-`
`Type __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UInt-`
`Type, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _U-`
`IntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _Random-`
`NumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _Random-`
`NumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution<`
`_IntType > &__x)`

- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType >`
`&__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType >`
`&__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType`
`> &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _Real-`
`Type > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _Real-`
`Type > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType >`
`&__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType >`
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution<`
`_RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, piecewise_linear_distribution< _RealType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::geometric_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::exponential_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::weibull_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::extreme_value_distribution< _RealType > & __x)`

5.421.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

Definition in file [bits/random.tcc](#).

5.422 random.tcc File Reference

Namespaces

- namespace [__gnu_cxx](#)

Macros

- `#define _EXT_RANDOM_TCC`

Functions

- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT,
_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::beta_distribution< _RealType > & __x)`

- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal-`
`_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution<`
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_-`
`distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution<`
`_RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _`
`RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_-`
`distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution<`
`_RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`
`bool __gnu_cxx::operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __`
`pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 >`
`& __lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1,`
`__sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > & __rhs)`
- `template<size_t _Dimen, typename _RealType >`
`bool __gnu_cxx::operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const`
`__gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT`
`, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_-`
`mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3,`
`__msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_-`
`distribution< _RealType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT,`
`_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_-`
`distribution< _Dimen, _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT,
_Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)`

5.422.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/random>`.

Definition in file [ext/random.tcc](#).

5.423 random_number.h File Reference

Classes

- class [__gnu_parallel::_RandomNumber](#)
Random number generator, based on the Mersenne twister.

Namespaces

- namespace [__gnu_parallel](#)

5.423.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random_number.h](#).

5.424 random_shuffle.h File Reference

Classes

- struct [__gnu_parallel::__DRandomShufflingGlobalData<_RAIter>](#)
Data known to every thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).
- struct [__gnu_parallel::__DRSSorterPU<_RAIter, _RandomNumberGenerator>](#)
Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Namespaces

- namespace [__gnu_parallel](#)

Typedefs

- typedef unsigned short [__gnu_parallel::_BinIndex](#)

Functions

- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle](#) (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle_drs](#) (_RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle_drs_pu](#) (_DRSSorterPU<_RAIter, _RandomNumberGenerator> * __pus)
- template<typename _RandomNumberGenerator>
int [__gnu_parallel::__random_number_pow2](#) (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Tp>
_Tp [__gnu_parallel::__round_up_to_pow2](#) (_Tp __x)
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__sequential_random_shuffle](#) (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)

5.424.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library. Definition in file [random_shuffle.h](#).

5.425 range_access.h File Reference

Namespaces

- namespace [std](#)

Functions

- `template<class _Container >`
`auto std::begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<class _Container >`
`auto std::begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm>`
`_Tp * std::begin (_Tp(&__arr)[_Nm])`
- `template<class _Container >`
`auto std::end (_Container &__cont) -> decltype(__cont.end())`
- `template<class _Container >`
`auto std::end (const _Container &__cont) -> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm>`
`_Tp * std::end (_Tp(&__arr)[_Nm])`

5.425.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [range_access.h](#).

5.426 ranged_hash_fn.hpp File Reference

Classes

- class [__gnu_pbds::detail::ranged_hash_fn](#)< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
- class [__gnu_pbds::detail::ranged_hash_fn](#)< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
- class [__gnu_pbds::detail::ranged_hash_fn](#)< Key, null_type, _Alloc, Comb_Hash_Fn, false >
- class [__gnu_pbds::detail::ranged_hash_fn](#)< Key, null_type, _Alloc, Comb_Hash_Fn, true >
- class [ranged_hash_fn](#)< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >

Primary template.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.426.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

Definition in file [ranged_hash_fn.hpp](#).

5.427 `ranged_probe_fn.hpp` File Reference

Classes

- class [__gnu_pbds::detail::ranged_probe_fn](#)< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
- class [__gnu_pbds::detail::ranged_probe_fn](#)< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
- class [__gnu_pbds::detail::ranged_probe_fn](#)< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
- class [ranged_probe_fn](#)< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >

Primary template.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.427.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probeing.

Definition in file [ranged_probe_fn.hpp](#).

5.428 `ratio` File Reference

Classes

- struct [std::ratio](#)< _Num, _Den >
Provides compile-time rational arithmetic.
- struct [std::ratio_equal](#)< _R1, _R2 >
ratio_equal
- struct [std::ratio_not_equal](#)< _R1, _R2 >
ratio_not_equal

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBCXX_RATIO`

Typedefs

- `template<typename _R1, typename _R2 >`
`using std::ratio_divide = typename __ratio_divide< _R1, _R2 >::type`
- `template<typename _R1, typename _R2 >`
`using std::ratio_multiply = typename __ratio_multiply< _R1, _R2 >::type`

5.428.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ratio](#).

5.429 ratio File Reference

Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

5.429.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file [tr2/ratio](#).

5.430 rb_tree File Reference

Classes

- struct [__gnu_cxx::rb_tree](#)< [_Key](#), [_Value](#), [_KeyOfValue](#), [_Compare](#), [_Alloc](#) >

Namespaces

- namespace [__gnu_cxx](#)

Macros

- `#define _RB_TREE`

5.430.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rb_tree](#).

5.431 rb_tree_.hpp File Reference

Classes

- class [__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >](#)
Red-Black tree.
This implementation uses an idea from the SGI STL (using a header node which is needed for efficient iteration).

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

5.431.1 Detailed Description

Contains an implementation for Red Black trees.

Definition in file [rb_tree_.hpp](#).

5.432 rc.hpp File Reference

Classes

- class [__gnu_pbds::detail::rc< _Node, _Alloc >](#)
Redundant binary counter.

Namespaces

- namespace [__gnu_pbds](#)

5.432.1 Detailed Description

Contains a redundant (binary counter).

Definition in file [rc.hpp](#).

5.433 rc_binomial_heap_.hpp File Reference

Classes

- class [__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RC_C_DEC`

5.433.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

Definition in file [rc_binomial_heap.hpp](#).

5.434 rc_string_base.h File Reference

Classes

- class [__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)

5.434.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file [rc_string_base.h](#).

5.435 regex File Reference

Macros

- `#define _GLIBCXX_REGEX`

5.435.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [regex](#).

5.436 regex.h File Reference

Classes

- class [std::basic_regex<_Ch_type, _Rx_traits>](#)

- class `std::match_results<_Bi_iter, _Alloc>`
The results of a match or search operation.
- class `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>`
- class `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>`
- struct `std::regex_traits<_Ch_type>`
Class `regex_traits`. Describes aspects of a regular expression.
- class `std::sub_match<_Biter>`

Namespaces

- namespace `std`

Typedefs

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`
`using std::__sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `typedef match_results< const char * > std::cmatch`
- `typedef regex_iterator< const char * > std::cregex_iterator`
- `typedef regex_token_iterator< const char * > std::cregex_token_iterator`
- `typedef sub_match< const char * > std::csub_match`
- `typedef basic_regex< char > std::regex`
- `typedef match_results< string::const_iterator > std::smatch`
- `typedef regex_iterator< string::const_iterator > std::sregex_iterator`
- `typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator`
- `typedef sub_match< string::const_iterator > std::ssub_match`
- `typedef match_results< const wchar_t * > std::wcmatch`
- `typedef regex_iterator< const wchar_t * > std::wcregex_iterator`
- `typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator`
- `typedef sub_match< const wchar_t * > std::wcs_sub_match`
- `typedef basic_regex< wchar_t > std::wregex`
- `typedef match_results< wstring::const_iterator > std::wsmatch`
- `typedef regex_iterator< wstring::const_iterator > std::wsregex_iterator`
- `typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator`
- `typedef sub_match< wstring::const_iterator > std::wssub_match`

Functions

- `template<typename _Bi_iter >`
`const sub_match< _Bi_iter > & std::__unmatched_sub ()`
- `template<typename _Bilter >`
`bool std::operator!= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, class _Alloc >`
`bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bilter >`
`bool std::operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bilter >`
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bilter >`
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`

- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

5.436.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex.h](#).

5.437 `regex_compiler.h` File Reference

Classes

- class `std::__detail::__Compiler< _InIter, _TraitsT >`
Builds an NFA from an input iterator interval.
- class `std::__detail::__Scanner< _InputIterator >`
`struct _Scanner.` *Scans an input range for regex tokens.*
- struct `std::__detail::__Scanner_base`
Base class for scanner.

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Functions

- `template<typename _InIter, typename _TraitsT > _AutomatonPtr std::__detail::__compile (const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)`

5.437.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex_compiler.h](#).

5.438 `regex_constants.h` File Reference

Namespaces

- namespace [std](#)
- namespace [std::regex_constants](#)

5.1 Regular Expression Syntax Options

- `enum std::regex_constants::__syntax_option { _S_icode, _S_nosubs, _S_optimize, _S_collate, _S_ECMAScript, _S_basic, _S_extended, _S_awk, _S_grep, _S_egrep, _S_syntax_last }`
- `typedef unsigned int std::regex_constants::syntax_option_type`
- `constexpr syntax_option_type std::regex_constants::icode`
- `constexpr syntax_option_type std::regex_constants::nosubs`
- `constexpr syntax_option_type std::regex_constants::optimize`
- `constexpr syntax_option_type std::regex_constants::collate`
- `constexpr syntax_option_type std::regex_constants::ECMAScript`
- `constexpr syntax_option_type std::regex_constants::basic`
- `constexpr syntax_option_type std::regex_constants::extended`
- `constexpr syntax_option_type std::regex_constants::awk`
- `constexpr syntax_option_type std::regex_constants::grep`
- `constexpr syntax_option_type std::regex_constants::egrep`

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `std::regex_constants::__match_flag` {
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- typedef `std::bitset`
`< _S_match_flag_last >` `std::regex_constants::match_flag_type`
- constexpr `match_flag_type` `std::regex_constants::match_default`
- constexpr `match_flag_type` `std::regex_constants::match_not_bol`
- constexpr `match_flag_type` `std::regex_constants::match_not_eol`
- constexpr `match_flag_type` `std::regex_constants::match_not_bow`
- constexpr `match_flag_type` `std::regex_constants::match_not_eow`
- constexpr `match_flag_type` `std::regex_constants::match_any`
- constexpr `match_flag_type` `std::regex_constants::match_not_null`
- constexpr `match_flag_type` `std::regex_constants::match_continuous`
- constexpr `match_flag_type` `std::regex_constants::match_prev_avail`
- constexpr `match_flag_type` `std::regex_constants::format_default`
- constexpr `match_flag_type` `std::regex_constants::format_sed`
- constexpr `match_flag_type` `std::regex_constants::format_no_copy`
- constexpr `match_flag_type` `std::regex_constants::format_first_only`

5.438.1 Detailed Description

Constant definitions for the std regex library. This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex_constants.h](#).

5.439 `regex_cursor.h` File Reference

Classes

- struct `std::__detail::PatternCursor`
ABC for pattern matching.
- class `std::__detail::SpecializedCursor<_FwdIterT>`
Provides a cursor into the specific target string.

Namespaces

- namespace `std`
- namespace `std::__detail`

Functions

- template<typename `_FwdIterT` >
`_SpecializedCursor<_FwdIterT>` `std::__detail::__cursor` (const `_FwdIterT` &`_b`, const `_FwdIterT` `_e`)

5.439.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex_cursor.h](#).

5.440 `regex_error.h` File Reference

Classes

- class [std::regex_error](#)
*A regular expression exception class.
The regular expression library throws objects of this class on error.*

Namespaces

- namespace [std](#)
- namespace [std::regex_constants](#)

Functions

- void [std::__throw_regex_error](#) (regex_constants::error_type __ecode)

5.3 Error Types

- enum [std::regex_constants::error_type](#) {
[_S_error_collate](#), [_S_error_ctype](#), [_S_error_escape](#), [_S_error_backref](#),
[_S_error_brack](#), [_S_error_paren](#), [_S_error_brace](#), [_S_error_badbrace](#),
[_S_error_range](#), [_S_error_space](#), [_S_error_badrepeat](#), [_S_error_complexity](#),
[_S_error_stack](#), [_S_error_last](#) }
• constexpr error_type [std::regex_constants::error_collate](#) (_S_error_collate)
• constexpr error_type [std::regex_constants::error_ctype](#) (_S_error_ctype)
• constexpr error_type [std::regex_constants::error_escape](#) (_S_error_escape)
• constexpr error_type [std::regex_constants::error_backref](#) (_S_error_backref)
• constexpr error_type [std::regex_constants::error_brack](#) (_S_error_brack)
• constexpr error_type [std::regex_constants::error_paren](#) (_S_error_paren)
• constexpr error_type [std::regex_constants::error_brace](#) (_S_error_brace)
• constexpr error_type [std::regex_constants::error_badbrace](#) (_S_error_badbrace)
• constexpr error_type [std::regex_constants::error_range](#) (_S_error_range)
• constexpr error_type [std::regex_constants::error_space](#) (_S_error_space)
• constexpr error_type [std::regex_constants::error_badrepeat](#) (_S_error_badrepeat)
• constexpr error_type [std::regex_constants::error_complexity](#) (_S_error_complexity)
• constexpr error_type [std::regex_constants::error_stack](#) (_S_error_stack)

5.440.1 Detailed Description

Error and exception objects for the std regex library. This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex_error.h](#).

5.441 `regex_grep_matcher.h` File Reference

Classes

- class `std::__detail::_Grep_matcher`
Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the `grep` utility, modified to use Laurikari tags.
- class `std::__detail::_SpecializedResults<_FwdIterT, _Alloc>`
A `_Results` facade specialized for wrapping a templated `match_results`.
- class `std::match_results<_Bi_iter, _Alloc>`
The results of a match or search operation.
- class `std::sub_match<_Biter>`

Namespaces

- namespace `std`
- namespace `std::__detail`

Typedefs

- typedef `std::stack<_StateIdT, std::vector<_StateIdT>>` `std::__detail::_StateStack`

5.441.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file `regex_grep_matcher.h`.

5.442 `regex_grep_matcher.tcc` File Reference

Namespaces

- namespace `std`
- namespace `std::__detail`

5.442.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file `regex_grep_matcher.tcc`.

5.443 `regex_nfa.h` File Reference

Classes

- class `std::__detail::_Automaton`

Base class for, um, automata. Could be an NFA or a DFA. Your choice.

- struct [std::__detail::CharMatcher<_InIterT, _TraitsT>](#)
Matches a single character.
- struct [std::__detail::EndTagger<_FwdIterT, _TraitsT>](#)
End state tag.
- class [std::__detail::_Nfa](#)
struct _Nfa
- struct [std::__detail::RangeMatcher<_InIterT, _TraitsT>](#)
Matches a character range (bracket expression)
- struct [std::__detail::Results](#)
Provides a generic facade for a templated match_results.
- struct [std::__detail::StartTagger<_FwdIterT, _TraitsT>](#)
Start state tag.
- struct [std::__detail::_State](#)
struct _State
- class [std::__detail::_StateSeq](#)
Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Typedefs

- typedef [std::shared_ptr](#)
[<_Automaton>](#) [std::__detail::_AutomatonPtr](#)
- typedef [std::function<bool\(const](#)
[_PatternCursor &\)>](#) [std::__detail::_Matcher](#)
- typedef [int](#) [std::__detail::_StateIdT](#)
- typedef [std::set<_StateIdT>](#) [std::__detail::_StateSet](#)
- typedef [std::function<void\(const](#)
[_PatternCursor &, _Results &\)>](#) [std::__detail::_Tagger](#)

Enumerations

- enum [std::__detail::_Opcode](#) {
 [_S_opcode_unknown](#), [_S_opcode_alternative](#), [_S_opcode_subexpr_begin](#), [_S_opcode_subexpr_end](#),
 [_S_opcode_match](#), [_S_opcode_accept](#) }

Functions

- [bool std::__detail::_AnyMatcher](#) (const [_PatternCursor](#) &)

Variables

- static const [_StateIdT](#) [std::__detail::_S_invalid_state_id](#)

5.443.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex_nfa.h](#).

5.444 `regex_nfa.tcc` File Reference

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

5.444.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex_nfa.tcc](#).

5.445 `resize_fn_imps.hpp` File Reference

5.445.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions.

Definition in file [cc_hash_table_map_/resize_fn_imps.hpp](#).

5.446 `resize_fn_imps.hpp` File Reference

5.446.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions.

Definition in file [gp_hash_table_map_/resize_fn_imps.hpp](#).

5.447 `resize_no_store_hash_fn_imps.hpp` File Reference

5.447.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is not stored.

Definition in file [cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp](#).

5.448 `resize_no_store_hash_fn_imps.hpp` File Reference

5.448.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is not stored.

Definition in file [gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp](#).

5.449 `resize_policy.hpp` File Reference

Classes

- class [__gnu_pbds::detail::resize_policy< _Tp >](#)
Resize policy for binary heap.

Namespaces

- namespace [__gnu_pbds](#)

5.449.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [resize_policy.hpp](#).

5.450 `resize_store_hash_fn_imps.hpp` File Reference

5.450.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is stored.

Definition in file [cc_hash_table_map_/resize_store_hash_fn_imps.hpp](#).

5.451 `resize_store_hash_fn_imps.hpp` File Reference

5.451.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is stored.

Definition in file [gp_hash_table_map_/resize_store_hash_fn_imps.hpp](#).

5.452 `rope` File Reference

Classes

- class [__gnu_cxx::rope< _CharT, _Alloc >](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [__gnu_cxx::__detail](#)
- namespace [std](#)
- namespace [std::tr1](#)

Macros

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`
- `#define _ROPE`

Typedefs

- `typedef rope< char > __gnu_cxx::crope`
- `typedef rope< wchar_t > __gnu_cxx::wrope`

Enumerations

- `enum { _S_max_rope_depth }`
- `enum _Tag { _S_leaf, _S_concat, _S_substringfn, _S_function }`

Functions

- `crope::reference __gnu_cxx::mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void __gnu_cxx::Destroy_const (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void __gnu_cxx::Destroy_const (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp >)`
- `template<class _CharT >`
`void __gnu_cxx::S_cond_store_eos (_CharT &)`
- `void __gnu_cxx::S_cond_store_eos (char &__c)`
- `void __gnu_cxx::S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`
`_CharT __gnu_cxx::S_eos (_CharT *)`
- `template<class _CharT >`
`bool __gnu_cxx::S_is_basic_char_type (_CharT *)`
- `bool __gnu_cxx::S_is_basic_char_type (char *)`
- `bool __gnu_cxx::S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`
`bool __gnu_cxx::S_is_one_byte_char_type (_CharT *)`
- `bool __gnu_cxx::S_is_one_byte_char_type (char *)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`

- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc >`
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

Variables

- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

5.452.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rope](#).

5.453 ropeimpl.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<class _CharT, class _Traits >`
`void __gnu_cxx::Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

5.453.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

Definition in file [ropeimpl.h](#).

5.454 rotate_fn_imps.hpp File Reference

5.454.1 Detailed Description

Contains imps for rotating nodes.

Definition in file [bin_search_tree_/rotate_fn_imps.hpp](#).

5.455 rotate_fn_imps.hpp File Reference

5.455.1 Detailed Description

Contains imps for rotating nodes.

Definition in file [pat_trie_/rotate_fn_imps.hpp](#).

5.456 safe_base.h File Reference

Classes

- class [__gnu_debug::_Safe_iterator_base](#)

Basic functionality for a safe iterator.

- class [__gnu_debug::__Safe_sequence_base](#)

Base class that supports tracking of iterators that reference a sequence.

Namespaces

- namespace [__gnu_debug](#)

5.456.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_base.h](#).

5.457 `safe_iterator.h` File Reference

Classes

- struct [__gnu_debug::BeforeBeginHelper<_Sequence>](#)
- class [__gnu_debug::Safe_iterator<_Iterator, _Sequence>](#)

Safe iterator wrapper.

Namespaces

- namespace [__gnu_debug](#)

Enumerations

- enum [__gnu_debug::Distance_precision](#) { [__dp_equality](#), [__dp_sign](#), [__dp_exact](#) }

Functions

- bool [__gnu_debug::__check_singular_aux](#) (const [_Safe_iterator_base](#) *__x)
- template<typename [_Iterator1](#) , typename [_Iterator2](#) >
[std::pair](#)< typename
[std::iterator_traits](#)
[<_Iterator1>](#)
[::difference_type](#),
[_Distance_precision](#) > [__gnu_debug::__get_distance](#) (const [_Iterator1](#) &__lhs, const [_Iterator2](#) &__rhs, [std::random_access_iterator_tag](#))
- template<typename [_Iterator1](#) , typename [_Iterator2](#) >
[std::pair](#)< typename
[std::iterator_traits](#)
[<_Iterator1>](#)
[::difference_type](#),
[_Distance_precision](#) > [__gnu_debug::__get_distance](#) (const [_Iterator1](#) &__lhs, const [_Iterator2](#) &__rhs, [std::forward_iterator_tag](#))

- `template<typename _Iterator1 , typename _Iterator2 >`
`std::pair< typename`
`std::iterator_traits`
`< _Iterator1 >`
`::difference_type,`
`_Distance_precision > __gnu_debug::__get_distance (const _Iterator1 &__lhs, const _Iterator2 &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`bool __gnu_debug::operator!= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<`
`_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`bool __gnu_debug::operator!= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<`
`_Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`_Safe_iterator< _Iterator,`
`_Sequence > __gnu_debug::operator+ (typename _Safe_iterator< _Iterator, _Sequence >::difference_type`
`__n, const _Safe_iterator< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`_Safe_iterator< _IteratorL,`
`_Sequence >::difference_type __gnu_debug::operator- (const _Safe_iterator< _IteratorL, _Sequence > &__-`
`lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`_Safe_iterator< _Iterator,`
`_Sequence >::difference_type __gnu_debug::operator- (const _Safe_iterator< _Iterator, _Sequence > &__lhs,`
`const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`bool __gnu_debug::operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<`
`_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`bool __gnu_debug::operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<`
`_Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`bool __gnu_debug::operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<`
`_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`bool __gnu_debug::operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<`
`_Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<`
`_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<`
`_Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<`
`_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<`
`_Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<`
`_IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<`
`_Iterator, _Sequence > &__rhs)`

5.457.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.h](#).

5.458 `safe_iterator.tcc` File Reference

Namespaces

- namespace [__gnu_debug](#)

Macros

- `#define _GLIBCXX_DEBUG_SAFE_ITERATOR_TCC`

5.458.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.tcc](#).

5.459 `safe_local_iterator.h` File Reference

Classes

- class [__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>](#)
Safe iterator wrapper.

Namespaces

- namespace [__gnu_debug](#)

Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>`
`bool __gnu_debug::operator!= (const _Safe_local_iterator<_IteratorL, _Sequence> &__lhs, const _Safe_local_iterator<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence>`
`bool __gnu_debug::operator!= (const _Safe_local_iterator<_Iterator, _Sequence> &__lhs, const _Safe_local_iterator<_Iterator, _Sequence> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>`
`bool __gnu_debug::operator== (const _Safe_local_iterator<_IteratorL, _Sequence> &__lhs, const _Safe_local_iterator<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence>`
`bool __gnu_debug::operator== (const _Safe_local_iterator<_Iterator, _Sequence> &__lhs, const _Safe_local_iterator<_Iterator, _Sequence> &__rhs)`

5.459.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_local_iterator.h](#).

5.460 `safe_local_iterator.tcc` File Reference

Namespaces

- namespace [__gnu_debug](#)

Macros

- `#define _GLIBCXX_DEBUG_SAFE_LOCAL_ITERATOR_TCC`

5.460.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_local_iterator.tcc](#).

5.461 `safe_sequence.h` File Reference

Classes

- class [__gnu_debug::After_nth_from<_Iterator>](#)
- class [__gnu_debug::Equal_to<_Type>](#)
- class [__gnu_debug::Not_equal_to<_Type>](#)
- class [__gnu_debug::Safe_iterator<_Iterator, _Sequence>](#)
Safe iterator wrapper.
- class [__gnu_debug::Safe_sequence<_Sequence>](#)
Base class for constructing a safe sequence type that tracks iterators that reference it.

Namespaces

- namespace [__gnu_debug](#)

5.461.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_sequence.h](#).

5.462 `safe_sequence.tcc` File Reference

Namespaces

- namespace [__gnu_debug](#)

Macros

- `#define _GLIBCXX_DEBUG_SAFE_SEQUENCE_TCC`

5.462.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_sequence.tcc](#).

5.463 `safe_unordered_base.h` File Reference

Classes

- class [__gnu_debug::__Safe_local_iterator_base](#)
Basic functionality for a safe iterator.
- class [__gnu_debug::__Safe_unordered_container_base](#)
Base class that supports tracking of local iterators that reference an unordered container.

Namespaces

- namespace [__gnu_debug](#)

5.463.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_unordered_base.h](#).

5.464 `safe_unordered_container.h` File Reference

Classes

- class [__gnu_debug::__Safe_unordered_container<_Container>](#)
Base class for constructing a safe unordered container type that tracks iterators that reference it.

Namespaces

- namespace [__gnu_debug](#)

5.464.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_unordered_container.h](#).

5.465 `safe_unordered_container.tcc` File Reference

Namespaces

- namespace [__gnu_debug](#)

Macros

- `#define _GLIBCXX_DEBUG_SAFE_UNORDERED_CONTAINER_TCC`

5.465.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_unordered_container.tcc](#).

5.466 `sample_probe_fn.hpp` File Reference

Classes

- class [__gnu_pbds::sample_probe_fn](#)
A sample probe policy.

Namespaces

- namespace [__gnu_pbds](#)

5.466.1 Detailed Description

Contains a sample probe policy.

Definition in file [sample_probe_fn.hpp](#).

5.467 `sample_range_hashing.hpp` File Reference

Classes

- class [__gnu_pbds::sample_range_hashing](#)
A sample range-hashing functor.

Namespaces

- namespace [__gnu_pbds](#)

5.467.1 Detailed Description

Contains a range hashing policy.

Definition in file [sample_range_hashing.hpp](#).

5.468 sample_ranged_hash_fn.hpp File Reference

Classes

- class [__gnu_pbds::sample_ranged_hash_fn](#)
A sample ranged-hash functor.

Namespaces

- namespace [__gnu_pbds](#)

5.468.1 Detailed Description

Contains a ranged hash policy.

Definition in file [sample_ranged_hash_fn.hpp](#).

5.469 sample_ranged_probe_fn.hpp File Reference

Classes

- class [__gnu_pbds::sample_ranged_probe_fn](#)
A sample ranged-probe functor.

Namespaces

- namespace [__gnu_pbds](#)

5.469.1 Detailed Description

Contains a ranged probe policy.

Definition in file [sample_ranged_probe_fn.hpp](#).

5.470 sample_resize_policy.hpp File Reference

Classes

- class [__gnu_pbds::sample_resize_policy](#)
A sample resize policy.

Namespaces

- namespace [__gnu_pbds](#)

5.470.1 Detailed Description

Contains a sample resize policy for hash tables.

Definition in file [sample_resize_policy.hpp](#).

5.471 `sample_resize_trigger.hpp` File Reference

Classes

- class [__gnu_pbds::sample_resize_trigger](#)
A sample resize trigger policy.

Namespaces

- namespace [__gnu_pbds](#)

5.471.1 Detailed Description

Contains a sample resize trigger policy class.

Definition in file [sample_resize_trigger.hpp](#).

5.472 `sample_size_policy.hpp` File Reference

Classes

- class [__gnu_pbds::sample_size_policy](#)
A sample size policy.

Namespaces

- namespace [__gnu_pbds](#)

5.472.1 Detailed Description

Contains a sample size resize-policy.

Definition in file [sample_size_policy.hpp](#).

5.473 `sample_tree_node_update.hpp` File Reference

Classes

- class [__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >](#)
A sample node updatator.

Namespaces

- namespace [__gnu_pbds](#)

5.473.1 Detailed Description

Contains a samle node update functor.

Definition in file [sample_tree_node_update.hpp](#).

5.474 `sample_trie_access_traits.hpp` File Reference

Classes

- struct [__gnu_pbds::sample_trie_access_traits](#)
A sample trie element access traits.

Namespaces

- namespace [__gnu_pbds](#)

5.474.1 Detailed Description

Contains a sample probe policy.

Definition in file [sample_trie_access_traits.hpp](#).

5.475 `sample_trie_node_update.hpp` File Reference

Classes

- class [__gnu_pbds::sample_trie_node_update](#)< [Node_Cltr](#), [Node_Itr](#), [_ATraits](#), [_Alloc](#) >
A sample node updatator.

Namespaces

- namespace [__gnu_pbds](#)

5.475.1 Detailed Description

Contains a samle node update functor.

Definition in file [sample_trie_node_update.hpp](#).

5.476 `sample_update_policy.hpp` File Reference

Classes

- struct [__gnu_pbds::sample_update_policy](#)
A sample list-update policy.

Namespaces

- namespace [__gnu_pbds](#)

5.476.1 Detailed Description

Contains a sample policy for list update containers.

Definition in file [sample_update_policy.hpp](#).

5.477 `scoped_allocator` File Reference

Classes

- class `std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >`
Primary class template.

Namespaces

- namespace `std`

Macros

- `#define _SCOPED_ALLOCATOR`

Functions

- `template<typename _Alloc >`
`auto std::__do_outermost (_Alloc &__a, _Alloc *) -> decltype(__a.outer_allocator())`
- `template<typename _Alloc >`
`_Alloc & std::__do_outermost (_Alloc &__a,...)`
- `template<typename _Alloc >`
`auto std::__outermost (_Alloc &__a) -> decltype(__do_outermost(__a,&__a))`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`
`bool std::operator!= (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`
`bool std::operator== (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept`

5.477.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `scoped_allocator`.

5.478 `search.h` File Reference

Namespaces

- namespace `__gnu_parallel`

Functions

- `template<typename _RAIter, typename _DifferenceTp >`
`void __gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp * __off)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`__RAIter1 __gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`

5.478.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [search.h](#).

5.479 set File Reference

Macros

- `#define _GLIBCXX_SET`

5.479.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [set](#).

5.480 set File Reference

Macros

- `#define _GLIBCXX_DEBUG_SET`

5.480.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set](#).

5.481 set File Reference

Macros

- `#define _GLIBCXX_PROFILE_SET`

5.481.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set](#).

5.482 set.h File Reference

Classes

- class [std::__debug::set<_Key, _Compare, _Allocator>](#)
Class `std::set` with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

5.482.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set.h](#).

5.483 set.h File Reference

Classes

- class [std::__profile::set< _Key, _Compare, _Allocator >](#)
Class std::set wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__profile::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

5.483.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set.h](#).

5.484 set_operations.h File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b, std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`
`_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`

5.484.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [set_operations.h](#).

5.485 settings.h File Reference

Classes

- struct [__gnu_parallel::_Settings](#)
class _Settings Run-time settings for the parallel mode including all tunable parameters.

Namespaces

- namespace [__gnu_parallel](#)

Macros

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

5.485.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

5.485.2 parallelization_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and ___off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort (__v.begin(), __v.end(), \_\_gnu\_parallel::sequential\_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. _It is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For `heuristic`, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

Definition in file [settings.h](#).

5.485.3 Macro Definition Documentation

5.485.3.1 `#define GLIBCXX_PARALLEL_CONDITION(__c)`

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

Parameters

<code>__c</code>	A condition that is convertible to <code>bool</code> that is overruled by <code>__gnu_parallel::_Settings::algorithm_</code> - strategy. Usually a decision based on the input size.
------------------	--

Definition at line 95 of file `settings.h`.

5.486 shared_ptr.h File Reference

Classes

- struct [owner_less< _Tp >](#)
Primary template `owner_less`.
- class [std::enable_shared_from_this< _Tp >](#)
Base class allowing use of member function `shared_from_this`.
- struct [std::hash< shared_ptr< _Tp > >](#)
`std::hash` specialization for `shared_ptr`.
- struct [std::owner_less< shared_ptr< _Tp > >](#)
Partial specialization of `owner_less` for `shared_ptr`.
- struct [std::owner_less< weak_ptr< _Tp > >](#)
Partial specialization of `owner_less` for `weak_ptr`.
- class [std::shared_ptr< _Tp >](#)
A smart pointer with reference-counted copy semantics.
- class [std::weak_ptr< _Tp >](#)
A smart pointer with weak semantics.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr<_Tp> std::allocate_shared (const _Alloc &__a, _Args &&...__args)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr<_Tp> std::const_pointer_cast (const shared_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr<_Tp> std::dynamic_pointer_cast (const shared_ptr<_Tp1> &__r) noexcept`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * std::get_deleter (const __shared_ptr<_Tp, _Lp> &__p) noexcept`
- `template<typename _Tp, typename... _Args>`
`shared_ptr<_Tp> std::make_shared (_Args &&...__args)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator!= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator!= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator< (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator< (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator< (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream<_Ch, _Tr> & std::operator<< (std::basic_ostream<_Ch, _Tr> &__os, const __shared_ptr<_Tp, _Lp> &__p)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator<= (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator<= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator<= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator== (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator> (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator> (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator> (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator>= (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr<_Tp> std::static_pointer_cast (const shared_ptr<_Tp1> &__r) noexcept`

- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

5.486.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [shared_ptr.h](#).

5.487 `shared_ptr_base.h` File Reference

Classes

- class [std::bad_weak_ptr](#)
Exception possibly thrown by `shared_ptr`.
- class [std::enable_shared_from_this< _Tp >](#)
Base class allowing use of member function `shared_from_this`.
- struct [std::hash< __shared_ptr< _Tp, _Lp > >](#)
`std::hash` specialization for `__shared_ptr`.
- class [std::shared_ptr< _Tp >](#)
A smart pointer with reference-counted copy semantics.
- class [std::weak_ptr< _Tp >](#)
A smart pointer with weak semantics.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp , _Lock_policy _Lp, typename _Alloc , typename... _Args>`
`__shared_ptr< _Tp, _Lp > std::__allocate_shared (const _Alloc &__a, _Args &&...__args)`
- `template<_Lock_policy _Lp, typename _Tp1 , typename _Tp2 >`
`void std::__enable_shared_from_this_helper (const __shared_count< _Lp > &, const __enable_shared_from_this< _Tp1, _Lp > *, const _Tp2 *) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`
`void std::__enable_shared_from_this_helper (const __shared_count<> &, const enable_shared_from_this< _Tp1 > *, const _Tp2 *) noexcept`
- `template<_Lock_policy _Lp>`
`void std::__enable_shared_from_this_helper (const __shared_count< _Lp > &,...) noexcept`
- `template<typename _Tp , _Lock_policy _Lp, typename... _Args>`
`__shared_ptr< _Tp, _Lp > std::__make_shared (_Args &&...__args)`
- `void std::__throw_bad_weak_ptr ()`
- `template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`

- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`

5.487.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [shared_ptr_base.h](#).

5.488 `size_fn_imps.hpp` File Reference

5.488.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container size related functions.

Definition in file [size_fn_imps.hpp](#).

5.489 `slice_array.h` File Reference

Classes

- class [std::slice](#)
Class defining one-dimensional subset of an array.
- class [std::slice_array<_Tp>](#)
Reference to one-dimensional subset of an array.

Namespaces

- namespace [std](#)

Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

5.489.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [slice_array.h](#).

5.490 `slist` File Reference

Classes

- class [__gnu_cxx::slist<_Tp, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Macros

- `#define _SLIST`

Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link` (`_Slist_node_base * __prev_node`, `_Slist_node_base * __new_node`)
- `_Slist_node_base * __gnu_cxx::__slist_previous` (`_Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `const _Slist_node_base * __gnu_cxx::__slist_previous` (`const _Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `_Slist_node_base * __gnu_cxx::__slist_reverse` (`_Slist_node_base * __node`)
- `size_t __gnu_cxx::__slist_size` (`_Slist_node_base * __node`)
- `void __gnu_cxx::__slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __before_first`, `_Slist_node_base * __before_last`)
- `void __gnu_cxx::__slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __head`)
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator!=` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator<` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator<=` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator==` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator>` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`
`bool __gnu_cxx::operator>=` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`
`void __gnu_cxx::swap` (`slist< _Tp, _Alloc > &__x`, `slist< _Tp, _Alloc > &__y`)

5.490.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [slist](#).

5.491 sort.h File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`
`void __gnu_parallel::__parallel_sort` (`_RAIter __begin`, `_RAIter __end`, `_Compare __comp`, `_Parallelism __parallelism`)

- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`

5.491.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [sort.h](#).

5.492 `splay_fn_imps.hpp` File Reference

5.492.1 Detailed Description

Contains an implementation class for `splay_tree_`.

Definition in file [splay_fn_imps.hpp](#).

5.493 `splay_tree_.hpp` File Reference

Classes

- class [__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >](#)
Splay tree.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node)`
- `#define PB_DS_CLASS_C_DEC`

- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_S_TREE_BASE`
- `#define PB_DS_S_TREE_BASE_NAME`
- `#define PB_DS_S_TREE_NAME`

5.493.1 Detailed Description

Contains an implementation class for splay trees.

Definition in file [splay_tree_.hpp](#).

5.494 `split_fn_imps.hpp` File Reference

5.494.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file [split_fn_imps.hpp](#).

5.495 `split_join_fn_imps.hpp` File Reference

5.495.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file [binary_heap_/split_join_fn_imps.hpp](#).

5.496 `split_join_fn_imps.hpp` File Reference

5.496.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file [binomial_heap_base_/split_join_fn_imps.hpp](#).

5.497 `split_join_fn_imps.hpp` File Reference

5.497.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file [bin_search_tree_/split_join_fn_imps.hpp](#).

5.498 `split_join_fn_imps.hpp` File Reference

5.498.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file [ov_tree_map_/split_join_fn_imps.hpp](#).

5.499 [split_join_fn_imps.hpp](#) File Reference

5.499.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file [pairing_heap_/split_join_fn_imps.hpp](#).

5.500 [split_join_fn_imps.hpp](#) File Reference

5.500.1 Detailed Description

Contains an implementation for `rb_tree_`.

Definition in file [rb_tree_map_/split_join_fn_imps.hpp](#).

5.501 [split_join_fn_imps.hpp](#) File Reference

5.501.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

Definition in file [rc_binomial_heap_/split_join_fn_imps.hpp](#).

5.502 [split_join_fn_imps.hpp](#) File Reference

5.502.1 Detailed Description

Contains an implementation class for `splay_tree_`.

Definition in file [splay_tree_/split_join_fn_imps.hpp](#).

5.503 [split_join_fn_imps.hpp](#) File Reference

5.503.1 Detailed Description

Contains an implementation for `thin_heap_`.

Definition in file [thin_heap_/split_join_fn_imps.hpp](#).

5.504 [sso_string_base.h](#) File Reference

Namespaces

- namespace [__gnu_cxx](#)

5.504.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file [sso_string_base.h](#).

5.505 sstream File Reference

Classes

- class [std::basic_istringstream< _CharT, _Traits, _Alloc >](#)
Controlling input for std::string.
- class [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#)
Controlling output for std::string.
- class [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#)
The actual work of input and output (for std::string).
- class [std::basic_stringstream< _CharT, _Traits, _Alloc >](#)
Controlling input and output for std::string.

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBCXX_SSTREAM`

5.505.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [sstream](#).

5.506 sstream.tcc File Reference

Namespaces

- namespace [std](#)

Macros

- `#define _SSTREAM_TCC`

5.506.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

Definition in file [sstream.tcc](#).

5.507 stack File Reference

Macros

- `#define _GLIBCXX_STACK`

5.507.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stack](#).

5.508 `standard_policies.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::default_comb_hash_fn](#)
Primary template, default_comb_hash_fn.
- struct [__gnu_pbds::detail::default_eq_fn< Key >](#)
Primary template, default_eq_fn.
- struct [__gnu_pbds::detail::default_hash_fn< Key >](#)
Primary template, default_hash_fn.
- struct [__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >](#)
Primary template, default_probe_fn.
- struct [__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >](#)
Primary template, default_resize_policy.
- struct [__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >](#)
Partial specialization, default_trie_access_traits.
- struct [__gnu_pbds::detail::default_update_policy](#)
Default update policy.
- struct [default_trie_access_traits< Key >](#)
Primary template, default_trie_access_traits.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

Enumerations

- enum { `default_store_hash` }

5.508.1 Detailed Description

Contains standard policies for containers.

Definition in file [standard_policies.hpp](#).

5.509 `stdc++.h` File Reference

5.509.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file [stdc++.h](#).

5.510 `stdexcept` File Reference

Classes

- class [std::domain_error](#)
- class [std::invalid_argument](#)
- class [std::length_error](#)
- class [std::logic_error](#)
One of two subclasses of exception.
- class [std::out_of_range](#)
- class [std::overflow_error](#)
- class [std::range_error](#)
- class [std::runtime_error](#)
One of two subclasses of exception.
- class [std::underflow_error](#)

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBCXX_STDEXCEPT`

5.510.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdexcept](#).

5.511 `stdio_filebuf.h` File Reference

Classes

- class [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#)
*Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

Namespaces

- namespace [__gnu_cxx](#)

5.511.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_filebuf.h](#).

5.512 `stdio_sync_filebuf.h` File Reference

Classes

- class [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#)
*Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

Namespaces

- namespace [__gnu_cxx](#)

5.512.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_sync_filebuf.h](#).

5.513 `stdtr1c++.h` File Reference

5.513.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file [stdtr1c++.h](#).

5.514 `stl_algo.h` File Reference

Namespaces

- namespace [std](#)

Enumerations

- enum { **_S_threshold** }
- enum { **_S_chunk_size** }

Functions

- template<typename _RandomAccessIterator, typename _Distance>
void **std::__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size)

- template<typename _RandomAccessIterator, typename _Distance, typename _Compare >
void **std::__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)
- template<typename _InputIterator, typename _Size, typename _OutputIterator >
_OutputIterator **std::__copy_n** (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >
_OutputIterator **std::__copy_n** (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)
- template<typename _RandomAccessIterator >
void **std::__final_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare >
void **std::__final_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _Tp >
_InputIterator **std::__find** (_InputIterator __first, _InputIterator __last, const _Tp &__val, input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Tp >
_RandomAccessIterator **std::__find** (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)
- template<typename _ForwardIterator1, typename _ForwardIterator2 >
_ForwardIterator1 **std::__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >
_ForwardIterator1 **std::__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >
_BidirectionalIterator1 **std::__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >
_BidirectionalIterator1 **std::__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _Predicate >
_InputIterator **std::__find_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Predicate >
_RandomAccessIterator **std::__find_if** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)
- template<typename _InputIterator, typename _Predicate >
_InputIterator **std::__find_if_not** (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Predicate >
_RandomAccessIterator **std::__find_if_not** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)
- template<typename _InputIterator, typename _Predicate >
_InputIterator **std::__find_if_not** (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate, typename _Distance >
_InputIterator **std::__find_if_not_n** (_InputIterator __first, _Distance &__len, _Predicate __pred)
- template<typename _EuclideanRingElement >
_EuclideanRingElement **std::__gcd** (_EuclideanRingElement __m, _EuclideanRingElement __n)
- template<typename _RandomAccessIterator >
void **std::__heap_select** (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare >
void **std::__heap_select** (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)

- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`
`_ForwardIterator std::__inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`
`void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsort (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::__introsort (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`
`void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`
`void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance >`
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer >`
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance >`
`void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`
`void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator >`
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _Iterator, typename _Compare >`
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _-`
`InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _-`
`InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _-`
`InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3 >`
`void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _-`
`BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`
`void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _-`
`BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward _-`
`iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`
`_BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __-`
`pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator _-`
`tag)`
- `template<typename _ForwardIterator >`
`void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator-`
`_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator`
`__last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`
`_BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, -`
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance`
`__buffer_size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp`
`& __val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp >`
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count,`
`const _Tp & __val, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp`
`& __val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count,`
`const _Tp & __val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`
`_ForwardIterator std::__stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate`
`__pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`

- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__unguarded_linear_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`
`_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`
`_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits`
`< _InputIterator >`
`::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits`
`< _InputIterator >`
`::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator,`
`_ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator,`
`_ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)`

- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator,`
`_FowardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator,`
`_FowardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccess-`
`Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccess-`
`Iterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccess-`
`Iterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccess-`
`Iterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1,`
`_OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true,`
`_OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumber-`
`Generator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`

- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

5.514.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

Definition in file [std_algo.h](#).

5.515 `std_algobase.h` File Reference

Classes

- struct `std::char_traits<_CharT>`
Basis for explicit traits specializations.

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`

Functions

- `template<bool _IsMove, typename _II, typename _OI>`
`_OI std::__copy_move_a(_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT>`
`__gnu_cxx::__enable_if`
`<__is_char<_CharT>::__value,`
`ostreambuf_iterator<_CharT,`
`char_traits<_CharT>`
`>::__type std::__copy_move_a2(_CharT *, _CharT *, ostreambuf_iterator<_CharT, char_traits<_CharT>`
`> >)`
- `template<bool _IsMove, typename _CharT>`
`__gnu_cxx::__enable_if`
`<__is_char<_CharT>::__value,`
`ostreambuf_iterator<_CharT,`
`char_traits<_CharT>`
`>::__type std::__copy_move_a2(const _CharT *, const _CharT *, ostreambuf_iterator<_CharT, char_`
`traits<_CharT> > >)`
- `template<bool _IsMove, typename _CharT>`
`__gnu_cxx::__enable_if`
`<__is_char<_CharT>::__value,`
`_CharT * >::__type std::__copy_move_a2(istreambuf_iterator<_CharT, char_traits<_CharT> > >,`
`istreambuf_iterator<_CharT, char_traits<_CharT> > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI>`
`_OI std::__copy_move_a2(_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`
`_BI2 std::__copy_move_backward_a(_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`
`_BI2 std::__copy_move_backward_a2(_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2>`
`bool std::__equal_aux(_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp>`
`__gnu_cxx::__enable_if`
`<!__is_scalar<_Tp>::__value,`
`void >::__type std::__fill_a(_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`

- `template<typename _Tp >`
`__gnu_cxx::__enable_if`
`< __is_byte< _Tp >::__value,`
`void >::__type std::__fill_a (_Tp * __first, _Tp * __last, const _Tp & __c)`
- `template<typename _OutputIterator , typename _Size , typename _Tp >`
`__gnu_cxx::__enable_if`
`< !__is_scalar< _Tp >::__value,`
`_OutputIterator >::__type std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp & __value)`
- `template<typename _Size , typename _Tp >`
`__gnu_cxx::__enable_if`
`< __is_byte< _Tp >::__value,`
`_Tp * >::__type std::__fill_n_a (_Tp * __first, _Size __n, const _Tp & __c)`
- `template<typename _II1 , typename _II2 >`
`bool std::__lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `constexpr int std::__lg (int __n)`
- `constexpr unsigned std::__lg (unsigned __n)`
- `constexpr long std::__lg (long __n)`
- `constexpr unsigned long std::__lg (unsigned long __n)`
- `constexpr long long std::__lg (long long __n)`
- `constexpr unsigned long long std::__lg (unsigned long long __n)`
- `template<typename _Iterator >`
`_Miter_base< _Iterator >`
`::iterator_type std::__miter_base (_Iterator __it)`
- `template<typename _Iterator >`
`_Niter_base< _Iterator >`
`::iterator_type std::__niter_base (_Iterator __it)`
- `template<typename _II , typename _OI >`
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1 , typename _BI2 >`
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >`
`bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1 , typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator , typename _Tp >`
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _OI , typename _Size , typename _Tp >`
`_OI std::fill_n (_OI __first, _Size __n, const _Tp & __value)`
- `template<typename _ForwardIterator1 , typename _ForwardIterator2 >`
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1 , typename _II2 >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1 , typename _II2 , typename _Compare >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _ForwardIterator , typename _Tp >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp , typename _Compare >`
`const _Tp & std::max (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp & __a, const _Tp & __b)`

- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1,`
`_InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1,`
`_InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Binary-`
`Predicate __binary_pred)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _-`
`__first2)`

5.515.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

Definition in file [std::algbase.h](#).

5.516 `std::bvector.h` File Reference

Classes

- struct [std::hash<::vector< bool, _Alloc > >](#)
std::hash specialization for vector<bool>.
- class [std::vector< bool, _Alloc >](#)
A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Namespaces

- namespace [std](#)

Typedefs

- typedef unsigned long [std::_Bit_type](#)

Enumerations

- enum { [_S_word_bit](#) }

Functions

- void [std::_fill_bvector](#) (_Bit_iterator __first, _Bit_iterator __last, bool __x)
- void [std::fill](#) (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)

- `_Bit_iterator` **`std::operator+`** (`ptrdiff_t __n`, `const _Bit_iterator &__x`)
- `_Bit_const_iterator` **`std::operator+`** (`ptrdiff_t __n`, `const _Bit_const_iterator &__x`)
- `ptrdiff_t` **`std::operator-`** (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)
- `void` **`std::swap`** (`_Bit_reference __x`, `_Bit_reference __y`) `noexcept`
- `void` **`std::swap`** (`_Bit_reference __x`, `bool &__y`) `noexcept`
- `void` **`std::swap`** (`bool &__x`, `_Bit_reference __y`) `noexcept`

5.516.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

Definition in file [stl_bvector.h](#).

5.517 `stl_construct.h` File Reference

Classes

- struct [std::allocator<_Tp>](#)
The standard allocator, as per [20.4].

Namespaces

- namespace [std](#)

Functions

- `template<typename _T1, typename... _Args>`
`void` [std::_Construct](#) (`_T1 *__p`, `_Args &&...__args`)
- `template<typename _Tp>`
`void` [std::_Destroy](#) (`_Tp *__pointer`)
- `template<typename _ForwardIterator>`
`void` [std::_Destroy](#) (`_ForwardIterator __first`, `_ForwardIterator __last`)
- `template<typename _ForwardIterator, typename _Allocator>`
`void` **`std::_Destroy`** (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Allocator &__alloc`)
- `template<typename _ForwardIterator, typename _Tp>`
`void` **`std::_Destroy`** (`_ForwardIterator __first`, `_ForwardIterator __last`, `allocator<_Tp> &`)

5.517.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [stl_construct.h](#).

5.518 `stl_deque.h` File Reference

Classes

- class [std::_Deque_base<_Tp, _Alloc>](#)

- struct `std::_Deque_iterator< _Tp, _Ref, _Ptr >`
A deque::iterator.
- class `std::deque< _Tp, _Alloc >`
A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

Functions

- `size_t std::__deque_buf_size (size_t __size)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::copy (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * >`
`__last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::copy_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &`
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<`
`_Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp *`
`> &__last, const _Tp &__value)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::move (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * >`
`__last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::move (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::move_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp`
`&, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp`
`&, _Tp * > std::move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<`
`_Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+ (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type std::operator- (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type std::operator- (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`

- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR,`
`_PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

5.518.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

Definition in file [std_deque.h](#).

5.518.2 Macro Definition Documentation

5.518.2.1 `#define GLIBCXX_DEQUE_BUF_SIZE`

This function controls the size of memory nodes.

Parameters

<code>__size</code>	The size of an element.
---------------------	-------------------------

Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 85 of file `std_deque.h`.

5.519 `std_function.h` File Reference

Classes

- struct [std::binary_function< _Arg1, _Arg2, _Result >](#)
- class [std::binary_negate< _Predicate >](#)
One of the [negation functors](#).
- class [std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >](#)
One of the [adaptors for member pointers](#).
- class [std::const_mem_fun1_t< _Ret, _Tp, _Arg >](#)
One of the [adaptors for member pointers](#).
- class [std::const_mem_fun_ref_t< _Ret, _Tp >](#)
One of the [adaptors for member pointers](#).
- class [std::const_mem_fun_t< _Ret, _Tp >](#)
One of the [adaptors for member pointers](#).

- struct `std::divides< _Tp >`
One of the [math functors](#).
- struct `std::equal_to< _Tp >`
One of the [comparison functors](#).
- struct `std::greater< _Tp >`
One of the [comparison functors](#).
- struct `std::greater_equal< _Tp >`
One of the [comparison functors](#).
- struct `std::less< _Tp >`
One of the [comparison functors](#).
- struct `std::less_equal< _Tp >`
One of the [comparison functors](#).
- struct `std::logical_and< _Tp >`
One of the [Boolean operations functors](#).
- struct `std::logical_not< _Tp >`
One of the [Boolean operations functors](#).
- struct `std::logical_or< _Tp >`
One of the [Boolean operations functors](#).
- class `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member pointers](#).
- class `std::mem_fun1_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member pointers](#).
- class `std::mem_fun_ref_t< _Ret, _Tp >`
One of the [adaptors for member pointers](#).
- class `std::mem_fun_t< _Ret, _Tp >`
One of the [adaptors for member pointers](#).
- struct `std::minus< _Tp >`
One of the [math functors](#).
- struct `std::modulus< _Tp >`
One of the [math functors](#).
- struct `std::multiplies< _Tp >`
One of the [math functors](#).
- struct `std::negate< _Tp >`
One of the [math functors](#).
- struct `std::not_equal_to< _Tp >`
One of the [comparison functors](#).
- struct `std::plus< _Tp >`
One of the [math functors](#).
- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
One of the [adaptors for function pointers](#).
- class `std::pointer_to_unary_function< _Arg, _Result >`
One of the [adaptors for function pointers](#).
- struct `std::unary_function< _Arg, _Result >`
- class `std::unary_negate< _Predicate >`
One of the [negation functors](#).

Namespaces

- namespace [std](#)

Functions

- `template<typename _Ret, typename _Tp >`
`mem_fun_t< _Ret, _Tp > std::mem_fun (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Predicate >`
`unary_negate< _Predicate > std::not1 (const _Predicate & __pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > std::not2 (const _Predicate & __pred)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function`
`< _Arg, _Result > std::ptr_fun (_Result(* __x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function`
`< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(* __x)(_Arg1, _Arg2))`

5.519.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [std_function.h](#).

5.520 `std_heap.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`

- `template<typename _RandomAccessIterator >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`
`void std::__push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`
`void std::__push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

5.520.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

Definition in file `std_heap.h`.

5.521 `std_iterator.h` File Reference

Classes

- class `std::back_insert_iterator<_Container>`
Turns assignment into insertion.
- class `std::front_insert_iterator<_Container>`
Turns assignment into insertion.
- class `std::insert_iterator<_Container>`
Turns assignment into insertion.
- class `std::move_iterator<_Iterator>`
- class `std::reverse_iterator<_Iterator>`

Namespaces

- namespace `__gnu_cxx`
- namespace `std`

Macros

- `#define _GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

Functions

- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`
`_ReturnType std::__make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Container>`
`back_insert_iterator<_Container> std::back_inserter (_Container &__x)`
- `template<typename _Container>`
`front_insert_iterator<_Container> std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator>`
`insert_iterator<_Container> std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator>`
`move_iterator<_Iterator> std::make_move_iterator (_Iterator __i)`
- `template<typename _Iterator>`
`bool std::operator!= (const reverse_iterator<_Iterator> &__x, const reverse_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR>`
`bool std::operator!= (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container>`
`bool __gnu_cxx::operator!= (const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs)`
- `template<typename _Iterator, typename _Container>`
`bool __gnu_cxx::operator!= (const __normal_iterator<_Iterator, _Container> &__lhs, const __normal_iterator<_Iterator, _Container> &__rhs)`
- `template<typename _IteratorL, typename _IteratorR>`
`bool std::operator!= (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y)`
- `template<typename _Iterator>`
`bool std::operator!= (const move_iterator<_Iterator> &__x, const move_iterator<_Iterator> &__y)`

- `template<typename _Iterator >`
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n,`
`const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator,`
`_Container > gnu_cxx::operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type`
`__n, const __normal_iterator< _Iterator, _Container > &__i)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const`
`move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >`
`::difference_type std::operator- (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) ->`
`decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`auto gnu_cxx::operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_-`
`iterator< _IteratorR, _Container > &__rhs) -> decltype(__lhs.base()-__rhs.base())`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator,`
`_Container >::difference_type gnu_cxx::operator- (const __normal_iterator< _Iterator, _Container > &__lhs,`
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) ->`
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`auto std::operator- (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) ->`
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool gnu_cxx::operator< (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_-`
`iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool gnu_cxx::operator< (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_-`
`iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool gnu_cxx::operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_-`
`iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool gnu_cxx::operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_-`
`iterator< _Iterator, _Container > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

5.521.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

Definition in file [stl_iterator.h](#).

5.522 `stl_iterator_base_funcs.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InputIterator, typename _Distance >`
`void std::advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`
`void std::advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`void std::advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _InputIterator >`
`iterator_traits`
`< _InputIterator >`
`::difference_type std::distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`iterator_traits`
`< _RandomAccessIterator >`
`::difference_type std::distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Distance >`
`void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator >`
`iterator_traits`
`< _InputIterator >`
`::difference_type std::distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::next (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`
`_BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)`

5.522.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

Definition in file [stl_iterator_base_funcs.h](#).

5.523 `std_iterator_base_types.h` File Reference

Classes

- class `std::__has_iterator_category_helper< _Tp >`
Traits class for iterators.
- struct `std::bidirectional_iterator_tag`
Bidirectional iterators support a superset of forward iterator operations.
- struct `std::forward_iterator_tag`
Forward iterators support a superset of input iterator operations.
- struct `std::input_iterator_tag`
Marking input iterators.
- struct `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
Common iterator class.
- struct `std::iterator_traits< _Tp * >`
Partial specialization for pointer types.
- struct `std::iterator_traits< const _Tp * >`
Partial specialization for const pointer types.
- struct `std::output_iterator_tag`
Marking output iterators.
- struct `std::random_access_iterator_tag`
Random-access iterators support a superset of bidirectional iterator operations.

Namespaces

- namespace `std`

Typedefs

- `template<typename _InIter >`
using `std::RequireInputIter` = `typename enable_if< is_convertible< typename iterator_traits< _InIter >::iterator_category, input_iterator_tag >::value >::type`

Functions

- `template<typename _Iter >`
`iterator_traits< _Iter >`
`::iterator_category` `std::__iterator_category` (`const _Iter &`)

5.523.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

Definition in file `std_iterator_base_types.h`.

5.524 `std::list.h` File Reference

Classes

- struct [std::__detail::_List_node_base](#)
Common part of a node in the list.
- class [std::_List_base<_Tp, _Alloc>](#)
See `bits/stl_deque.h`'s `_Deque_base` for an explanation.
- struct [std::_List_const_iterator<_Tp>](#)
A `list::const_iterator`.
- struct [std::_List_iterator<_Tp>](#)
A `list::iterator`.
- struct [std::_List_node<_Tp>](#)
An actual node in the list.
- class [std::list<_Tp, _Alloc>](#)
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Functions

- `template<typename _Val >`
`bool std::operator!= (const _List_iterator<_Val> &__x, const _List_const_iterator<_Val> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Val >`
`bool std::operator== (const _List_iterator<_Val> &__x, const _List_const_iterator<_Val> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (list<_Tp, _Alloc> &__x, list<_Tp, _Alloc> &__y)`

5.524.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

Definition in file [std::list.h](#).

5.525 `std::map.h` File Reference

Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Namespaces

- namespace `std`

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`

5.525.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

Definition in file `std::map.h`.

5.526 `std::multimap.h` File Reference

Classes

- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Namespaces

- namespace `std`

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

5.526.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

Definition in file [std_multimap.h](#).

5.527 `std_multiset.h` File Reference

Classes

- class [std::multiset< _Key, _Compare, _Alloc >](#)
A standard container made up of elements, which can be retrieved in logarithmic time.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

5.527.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

Definition in file [std::multiset.h](#).

5.528 `std::numeric.h` File Reference

Namespaces

- namespace `std`

Functions

- `template<typename _InputIterator, typename _Tp >`
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`_Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`
`_Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

5.528.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

Definition in file [std_numeric.h](#).

5.529 `std::pair.h` File Reference

Classes

- struct [std::pair<_T1, _T2>](#)
Struct holding two objects of arbitrary type.
- struct [std::piecewise_construct_t](#)
piecewise_construct_t
- class [std::tuple<_Elements>](#)
Primary class template, tuple.

Namespaces

- namespace [std](#)

Functions

- `template<class _T1, class _T2>`
`constexpr pair< typename`
`__decay_and_strip< _T1 >`
`::__type, typename`
`__decay_and_strip< _T2 >`
`::__type> std::make_pair (_T1 &&__x, _T2 &&__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2>`
`void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`

Variables

- `constexpr piecewise_construct_t std::piecewise_construct`

5.529.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Definition in file [std::pair.h](#).

5.530 `std::queue.h` File Reference

Classes

- class [std::priority_queue<_Tp, _Sequence, _Compare>](#)
A standard container automatically sorting its contents.
- class [std::queue<_Tp, _Sequence>](#)
A standard container giving FIFO behavior.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Seq>`
`bool std::operator!= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`
`bool std::operator< (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`
`bool std::operator<= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`
`bool std::operator== (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`
`bool std::operator> (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`
`bool std::operator>= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`
`void std::swap (queue<_Tp, _Seq> &__x, queue<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Sequence, typename _Compare>`
`void std::swap (priority_queue<_Tp, _Sequence, _Compare> &__x, priority_queue<_Tp, _Sequence, _Compare> &__y) noexcept(noexcept(__x.swap(__y)))`

5.530.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

Definition in file [std::queue.h](#).

5.531 `std::raw_storage_iter.h` File Reference

Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp>`

Namespaces

- namespace `std`

5.531.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file `std::raw_storage_iter.h`.

5.532 `std::rel_ops.h` File Reference

Namespaces

- namespace `std`
- namespace `std::rel_ops`

Functions

- template<class `_Tp`>
bool `std::rel_ops::operator!=` (const `_Tp` &__x, const `_Tp` &__y)
- template<class `_Tp`>
bool `std::rel_ops::operator<=` (const `_Tp` &__x, const `_Tp` &__y)
- template<class `_Tp`>
bool `std::rel_ops::operator>` (const `_Tp` &__x, const `_Tp` &__y)
- template<class `_Tp`>
bool `std::rel_ops::operator>=` (const `_Tp` &__x, const `_Tp` &__y)

5.532.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std::utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads

Short summary: the `rel_ops` operators should be avoided for the present.

Definition in file `std::rel_ops.h`.

5.533 `std::set.h` File Reference

Classes

- class `std::set<_Key, _Compare, _Alloc>`
A standard container made up of unique keys, which can be retrieved in logarithmic time.

Namespaces

- namespace `std`

Functions

- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator!= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator< (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator<= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator== (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator> (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator>= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`void std::swap (set<_Key, _Compare, _Alloc> &__x, set<_Key, _Compare, _Alloc> &__y)`

5.533.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

Definition in file `std::set.h`.

5.534 `std::stack.h` File Reference

Classes

- class `std::stack<_Tp, _Sequence>`
A standard container giving FILO behavior.

Namespaces

- namespace `std`

Functions

- `template<typename _Tp, typename _Seq >`
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`void std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

5.534.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

Definition in file [std::stack.h](#).

5.535 `std::tempbuf.h` File Reference

Classes

- class [std::_Temporary_buffer<_ForwardIterator, _Tp >](#)

Namespaces

- namespace [std](#)

Functions

- `template<typename _Pointer, typename _ForwardIterator >`
`void std::__uninitialized_construct_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _Tp >`
`pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp >`
`void std::return_temporary_buffer (_Tp *__p)`

5.535.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [std::tempbuf.h](#).

5.536 `std::tree.h` File Reference

Namespaces

- namespace [std](#)

Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

Functions

- unsigned int `std::Rb_tree_black_count` (const `_Rb_tree_node_base` * __node, const `_Rb_tree_node_base` * __root) throw ()
- `_Rb_tree_node_base` * `std::Rb_tree_decrement` (`_Rb_tree_node_base` * __x) throw ()
- const `_Rb_tree_node_base` * `std::Rb_tree_decrement` (const `_Rb_tree_node_base` * __x) throw ()
- `_Rb_tree_node_base` * `std::Rb_tree_increment` (`_Rb_tree_node_base` * __x) throw ()
- const `_Rb_tree_node_base` * `std::Rb_tree_increment` (const `_Rb_tree_node_base` * __x) throw ()
- void `std::Rb_tree_insert_and_rebalance` (const bool __insert_left, `_Rb_tree_node_base` * __x, `_Rb_tree_node_base` * __p, `_Rb_tree_node_base` & __header) throw ()
- `_Rb_tree_node_base` * `std::Rb_tree_rebalance_for_erase` (`_Rb_tree_node_base` * const __z, `_Rb_tree_node_base` & __header) throw ()
- template<typename `_Val` >
bool `std::operator!=` (const `_Rb_tree_iterator`< `_Val` > & __x, const `_Rb_tree_const_iterator`< `_Val` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator!=` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator<` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator<=` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)
- template<typename `_Val` >
bool `std::operator==` (const `_Rb_tree_iterator`< `_Val` > & __x, const `_Rb_tree_const_iterator`< `_Val` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator==` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator>` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator>=` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
void `std::swap` (`_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __x, `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & __y)

5.536.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

Definition in file [std_tree.h](#).

5.537 `std_uninitialized.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __-`
`result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __-`
`result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_copy_move (_InputIterator1 __first1, _InputIterator1 __last1, _Input-`
`Iterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input-`
`_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __-`
`result, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void std::uninitialized_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void std::uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`
`void std::uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`
`void std::uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void std::uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator`
`&__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void std::uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator<`
`_Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp`
`&__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`

- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void std::__uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`

5.537.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [std_uninitialized.h](#).

5.538 `std_vector.h` File Reference

Classes

- struct [std::Vector_base< _Tp, _Alloc >](#)
See `bits/stl_deque.h`'s `_Deque_base` for an explanation.
- class [std::vector< _Tp, _Alloc >](#)
A standard container which offers fixed time access to individual elements in any order.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`

5.538.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

Definition in file [std_vector.h](#).

5.539 `stream_iterator.h` File Reference

Classes

- class [std::istream_iterator< _Tp, _CharT, _Traits, _Dist >](#)
Provides input iterator semantics for streams.
- class [std::ostream_iterator< _Tp, _CharT, _Traits >](#)
Provides output iterator semantics for streams.

Namespaces

- namespace [std](#)

Functions

- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`

5.539.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [stream_iterator.h](#).

5.540 streambuf File Reference

Classes

- class [std::basic_streambuf<_CharT, _Traits>](#)
The actual work of input and output (interface).

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBXX_STREAMBUF`

Functions

- `template<typename _CharT, typename _Traits>`
streamsize **std::__copy_streambufs_eof** (basic_streambuf<_CharT, _Traits> *, basic_streambuf<_CharT, _Traits> *, bool &)
- `template<>`
streamsize **std::__copy_streambufs_eof** (basic_streambuf<char> *__sbin, basic_streambuf<char> *__sbout, bool &__ineof)
- `template<>`
streamsize **std::__copy_streambufs_eof** (basic_streambuf<wchar_t> *__sbin, basic_streambuf<wchar_t> *__sbout, bool &__ineof)

5.540.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [streambuf](#).

5.541 streambuf.tcc File Reference

Namespaces

- namespace [std](#)

Macros

- `#define _STREAMBUF_TCC`

Functions

- `template<typename _CharT, typename _Traits>`
streamsize **std::__copy_streambufs** (basic_streambuf<_CharT, _Traits> *__sbin, basic_streambuf<_CharT, _Traits> *__sbout)

- template<typename _CharT, typename _Traits >
streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)

5.541.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<streambuf>`.

Definition in file [streambuf.tcc](#).

5.542 streambuf_iterator.h File Reference

Classes

- class [std::istreambuf_iterator< _CharT, _Traits >](#)
Provides input iterator semantics for streambufs.
- class [std::ostreambuf_iterator< _CharT, _Traits >](#)
Provides output iterator semantics for streambufs.

Namespaces

- namespace [std](#)

Functions

- template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if
< __is_char< _CharT >::__value,
ostreambuf_iterator< _CharT >
>::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if
< __is_char< _CharT >::__value,
ostreambuf_iterator< _CharT >
>::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if
< __is_char< _CharT >::__value,
_CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _CharT >
__gnu_cxx::__enable_if
< __is_char< _CharT >::__value,
ostreambuf_iterator< _CharT >
>::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)

- `template<typename _CharT >`
`__gnu_cxx::__enable_if`
`< __is_char< _CharT >::__value,`
`istreambuf_iterator< _CharT >`
`>::__type std::find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT`
`&__val)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`
`_Traits > &__b)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`
`_Traits > &__b)`

5.542.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [streambuf_iterator.h](#).

5.543 string File Reference

Macros

- `#define _GLIBCXX_STRING`

5.543.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [string](#).

5.544 string File Reference

Classes

- class [__gnu_debug::basic_string< _CharT, _Traits, _Allocator >](#)
Class `std::basic_string` with safety/checking/debug instrumentation.

Namespaces

- namespace [__gnu_debug](#)

Macros

- `#define _GLIBCXX_DEBUG_STRING`

Typedefs

- typedef basic_string< char > **__gnu_debug::string**
- typedef basic_string< wchar_t > **__gnu_debug::wstring**

Functions

- template<typename _CharT, typename _Traits, typename _Allocator >
[std::basic_istream](#)< _CharT,
 _Traits > & **__gnu_debug::getline** ([std::basic_istream](#)< _CharT, _Traits > &__is, basic_string< _CharT, _Traits,
 _Allocator > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Allocator >
[std::basic_istream](#)< _CharT,
 _Traits > & **__gnu_debug::getline** ([std::basic_istream](#)< _CharT, _Traits > &__is, basic_string< _CharT, _Traits,
 _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator >
 bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
 _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 bool **__gnu_debug::operator!=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__ -
 rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__ -
 rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 basic_string< _CharT, _Traits,
 _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic-
 string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 basic_string< _CharT, _Traits,
 _Allocator > **__gnu_debug::operator+** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator
 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 basic_string< _CharT, _Traits,
 _Allocator > **__gnu_debug::operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__ -
 rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 basic_string< _CharT, _Traits,
 _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT
 *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 basic_string< _CharT, _Traits,
 _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __ -
 rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
 _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 bool **__gnu_debug::operator<** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__ -
 rhs)
- template<typename _CharT, typename _Traits, typename _Allocator >
 bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__ -
 rhs)

- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_ostream< _CharT,`
`_Traits > & __gnu_debug::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const basic_string<`
`_CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<`
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_istream< _CharT,`
`_Traits > & __gnu_debug::operator>> (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT,`
`_Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits,`
`_Allocator > &__rhs)`

5.544.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/string](#).

5.545 string_conversions.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>
_Ret __gnu_cxx::__stoa (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const
_CharT * __str, std::size_t * __idx, _Base... __base)`
- `template<typename _String, typename _CharT = typename _String::value_type>
_String __gnu_cxx::__to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std-
::size_t __n, const _CharT * __fmt,...)`

5.545.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [string_conversions.h](#).

5.546 stringfwd.h File Reference

Classes

- struct [std::allocator< _Tp >](#)
The standard allocator, as per [20.4].
- class [std::basic_string< _CharT, _Traits, _Alloc >](#)
Managing sequences of characters and character-like objects.
- struct [std::char_traits< _CharT >](#)
Basis for explicit traits specializations.

Namespaces

- namespace [std](#)

Typedefs

- `typedef basic_string< char > std::string`
- `typedef basic_string< char16_t > std::u16string`
- `typedef basic_string< char32_t > std::u32string`
- `typedef basic_string< wchar_t > std::wstring`

5.546.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [stringfwd.h](#).

5.547 **strstream** File Reference

Namespaces

- namespace [std](#)

5.547.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

Definition in file [strstream](#).

5.548 **synth_access_traits.hpp** File Reference

Classes

- struct [__gnu_pbds::detail::synth_access_traits](#)< [Type_Traits](#), [Set](#), [_ATraits](#) >
Synthetic element access traits.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC`
- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC`

5.548.1 Detailed Description

Contains an implementation class for a patricia tree.

Definition in file [synth_access_traits.hpp](#).

5.549 **system_error** File Reference

Classes

- class [std::error_category](#)
error_category
- struct [std::error_code](#)
error_code
- struct [std::error_condition](#)
error_condition
- struct [std::hash](#)< [error_code](#) >
std::hash specialization for error_code.
- struct [std::is_error_code_enum](#)< [_Tp](#) >
is_error_code_enum

- struct [std::is_error_condition_enum< _Tp >](#)
is_error_condition_enum
- class [std::system_error](#)
Thrown to indicate error code of underlying system.

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBCXX_SYSTEM_ERROR`

Functions

- `const error_category & std::generic_category () noexcept`
- `error_code std::make_error_code (errc __e) noexcept`
- `error_condition std::make_error_condition (errc __e) noexcept`
- `bool std::operator!= (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator!= (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `bool std::operator!= (const error_condition &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator!= (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `bool std::operator< (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator< (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const error-
_code &__e)`
- `bool std::operator== (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator== (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `bool std::operator== (const error_condition &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `const error_category & std::system_category () noexcept`

Variables

- `error_code std::make_error_code (errc) noexcept`
- `error_condition std::make_error_condition (errc) noexcept`

5.549.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [system_error](#).

5.550 tag_and_trait.hpp File Reference

Classes

- struct [__gnu_pbds::associative_tag](#)
Basic associative-container.
- struct [__gnu_pbds::basic_branch_tag](#)
Basic branch structure.
- struct [__gnu_pbds::basic_hash_tag](#)
Basic hash structure.
- struct [__gnu_pbds::basic_invalidation_guarantee](#)
- struct [__gnu_pbds::binary_heap_tag](#)
Binary-heap (array-based).
- struct [__gnu_pbds::binomial_heap_tag](#)
Binomial-heap.
- struct [__gnu_pbds::cc_hash_tag](#)
Collision-chaining hash.
- struct [__gnu_pbds::container_tag](#)
Base data structure tag.
- struct [__gnu_pbds::container_traits< Cntnr >](#)
Container traits.
- struct [__gnu_pbds::container_traits_base< binary_heap_tag >](#)
Specialization, binary heap.
- struct [__gnu_pbds::container_traits_base< binomial_heap_tag >](#)
Specialization, binomial heap.
- struct [__gnu_pbds::container_traits_base< cc_hash_tag >](#)
Specialization, cc hash.
- struct [__gnu_pbds::container_traits_base< gp_hash_tag >](#)
Specialization, gp hash.
- struct [__gnu_pbds::container_traits_base< list_update_tag >](#)
Specialization, list update.
- struct [__gnu_pbds::container_traits_base< ov_tree_tag >](#)
Specialization, ov tree.
- struct [__gnu_pbds::container_traits_base< pairing_heap_tag >](#)
Specialization, pairing heap.
- struct [__gnu_pbds::container_traits_base< pat_trie_tag >](#)
Specialization, pat trie.
- struct [__gnu_pbds::container_traits_base< rb_tree_tag >](#)
Specialization, rb tree.
- struct [__gnu_pbds::container_traits_base< rc_binomial_heap_tag >](#)
Specialization, rc binomial heap.
- struct [__gnu_pbds::container_traits_base< splay_tree_tag >](#)
Specialization, splay tree.
- struct [__gnu_pbds::container_traits_base< thin_heap_tag >](#)
Specialization, thin heap.
- struct [__gnu_pbds::gp_hash_tag](#)
General-probing hash.
- struct [__gnu_pbds::list_update_tag](#)

List-update.

- struct [__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >](#)
A null node updatator, indicating that no node updates are required.
- struct [__gnu_pbds::null_type](#)
Represents no type, or absence of type, for template tricks.
- struct [__gnu_pbds::ov_tree_tag](#)
Ordered-vector tree.
- struct [__gnu_pbds::pairing_heap_tag](#)
Pairing-heap.
- struct [__gnu_pbds::pat_trie_tag](#)
PATRICIA trie.
- struct [__gnu_pbds::point_invalidation_guarantee](#)
- struct [__gnu_pbds::priority_queue_tag](#)
Basic priority-queue.
- struct [__gnu_pbds::range_invalidation_guarantee](#)
- struct [__gnu_pbds::rb_tree_tag](#)
Red-black tree.
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
Redundant-counter binomial-heap.
- struct [__gnu_pbds::sequence_tag](#)
Basic sequence.
- struct [__gnu_pbds::splay_tree_tag](#)
Splay tree.
- struct [__gnu_pbds::string_tag](#)
Basic string container, inclusive of strings, ropes, etc.
- struct [__gnu_pbds::thin_heap_tag](#)
Thin heap.
- struct [__gnu_pbds::tree_tag](#)
Basic tree structure.
- struct [__gnu_pbds::trie_tag](#)
Basic trie structure.
- struct [__gnu_pbds::trivial_iterator_tag](#)
A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.
- struct [container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >](#)
Dispatch mechanism, primary template for associative types.
- struct [container_traits_base< _Tag >](#)
Primary template, container traits base.

Namespaces

- namespace [__gnu_pbds](#)

Typedefs

- typedef void [__gnu_pbds::trivial_iterator_difference_type](#)

5.550.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file [tag_and_trait.hpp](#).

5.551 tags.h File Reference

Classes

- struct [__gnu_parallel::balanced_quicksort_tag](#)
Forces parallel sorting using balanced quicksort at compile time.
- struct [__gnu_parallel::balanced_tag](#)
Recommends parallel execution using dynamic load-balancing at compile time.
- struct [__gnu_parallel::constant_size_blocks_tag](#)
Selects the constant block size variant for `std::find()`.
- struct [__gnu_parallel::default_parallel_tag](#)
Recommends parallel execution using the default parallel algorithm.
- struct [__gnu_parallel::equal_split_tag](#)
Selects the equal splitting variant for `std::find()`.
- struct [__gnu_parallel::exact_tag](#)
Forces parallel merging with exact splitting, at compile time.
- struct [__gnu_parallel::find_tag](#)
Base class for `std::find()` variants.
- struct [__gnu_parallel::growing_blocks_tag](#)
Selects the growing block size variant for `std::find()`.
- struct [__gnu_parallel::multiway_mergesort_exact_tag](#)
Forces parallel sorting using multiway mergesort with exact splitting at compile time.
- struct [__gnu_parallel::multiway_mergesort_sampling_tag](#)
Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.
- struct [__gnu_parallel::multiway_mergesort_tag](#)
Forces parallel sorting using multiway mergesort at compile time.
- struct [__gnu_parallel::omp_loop_static_tag](#)
Recommends parallel execution using OpenMP static load-balancing at compile time.
- struct [__gnu_parallel::omp_loop_tag](#)
Recommends parallel execution using OpenMP dynamic load-balancing at compile time.
- struct [__gnu_parallel::parallel_tag](#)
Recommends parallel execution at compile time, optionally using a user-specified number of threads.
- struct [__gnu_parallel::quicksort_tag](#)
Forces parallel sorting using unbalanced quicksort at compile time.
- struct [__gnu_parallel::sampling_tag](#)
Forces parallel merging with exact splitting, at compile time.
- struct [__gnu_parallel::sequential_tag](#)
Forces sequential execution at compile time.
- struct [__gnu_parallel::unbalanced_tag](#)
Recommends parallel execution using static load-balancing at compile time.

Namespaces

- namespace [__gnu_parallel](#)

5.551.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [tags.h](#).

5.552 `tgmath.h` File Reference

Macros

- `#define _GLIBCXX_TGMATH_H`

5.552.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tgmath.h](#).

5.553 `thin_heap_.hpp` File Reference

Classes

- class [__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

Enumerations

- enum { `num_distinct_rank_bounds` }

Variables

- static const std::size_t [__gnu_pbds::detail::g_a_rank_bounds](#) [`num_distinct_rank_bounds`]

5.553.1 Detailed Description

Contains an implementation class for a thin heap.

Definition in file [thin_heap_.hpp](#).

5.554 thread File Reference

Classes

- struct [std::hash< thread::id >](#)
std::hash specialization for thread::id.
- class [std::thread](#)
thread
- class [std::thread::id](#)
thread::id

Namespaces

- namespace [std](#)
- namespace [std::this_thread](#)

Macros

- `#define _GLIBCXX_THREAD`

Functions

- void [std::this_thread::__sleep_for](#) (chrono::seconds, chrono::nanoseconds)
- thread::id [std::this_thread::get_id](#) () noexcept
- bool [std::operator!=](#) (thread::id __x, thread::id __y) noexcept
- template<class _CharT, class _Traits >
basic_ostream< _CharT, _Traits > & [std::operator<<](#) (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- bool [std::operator<=](#) (thread::id __x, thread::id __y) noexcept
- bool [std::operator>](#) (thread::id __x, thread::id __y) noexcept
- bool [std::operator>=](#) (thread::id __x, thread::id __y) noexcept
- template<typename _Rep, typename _Period >
void [std::this_thread::sleep_for](#) (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration >
void [std::this_thread::sleep_until](#) (const chrono::time_point< _Clock, _Duration > &__atime)
- void [std::swap](#) (thread &__x, thread &__y) noexcept
- void [std::this_thread::yield](#) () noexcept

5.554.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [thread](#).

5.555 throw_allocator.h File Reference

Classes

- struct [__gnu_cxx::annotate_base](#)
Base class for checking address and label information about allocations. Create a std::map between the allocated address (void) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*
- struct [__gnu_cxx::condition_base](#)
Base struct for condition policy.
- struct [__gnu_cxx::forced_error](#)
Thrown by exception safety machinery.
- struct [__gnu_cxx::limit_condition](#)
Base class for incremental control and throw.
- struct [__gnu_cxx::limit_condition::always_adjustor](#)
Always enter the condition.
- struct [__gnu_cxx::limit_condition::limit_adjustor](#)
Enter the nth condition.
- struct [__gnu_cxx::limit_condition::never_adjustor](#)
Never enter the condition.
- struct [__gnu_cxx::random_condition](#)
Base class for random probability control and throw.
- struct [__gnu_cxx::random_condition::always_adjustor](#)
Always enter the condition.
- struct [__gnu_cxx::random_condition::group_adjustor](#)
Group condition.
- struct [__gnu_cxx::random_condition::never_adjustor](#)
Never enter the condition.
- class [__gnu_cxx::throw_allocator_base< _Tp, _Cond >](#)
*Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templated code.
Note: Deallocate not allowed to throw.*
- struct [__gnu_cxx::throw_allocator_limit< _Tp >](#)
Allocator throwing via limit condition.
- struct [__gnu_cxx::throw_allocator_random< _Tp >](#)
Allocator throwing via random condition.
- struct [__gnu_cxx::throw_value_base< _Cond >](#)
Class with exception generation control. Intended to be used as a value_type in templated code.
- struct [__gnu_cxx::throw_value_limit](#)
Type throwing via limit condition.
- struct [__gnu_cxx::throw_value_random](#)
Type throwing via random condition.
- struct [std::hash< __gnu_cxx::throw_value_limit >](#)
Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.
- struct [std::hash< __gnu_cxx::throw_value_random >](#)
Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- void **__gnu_cxx::__throw_forced_error** ()
- template<typename _Tp, typename _Cond >
bool **__gnu_cxx::operator!=** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
throw_value_base< _Cond > **__gnu_cxx::operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
throw_value_base< _Cond > **__gnu_cxx::operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
throw_value_base< _Cond > **__gnu_cxx::operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
bool **__gnu_cxx::operator<** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- [std::ostream](#) & **__gnu_cxx::operator<<** ([std::ostream](#) &os, const annotate_base &__b)
- template<typename _Cond >
bool **__gnu_cxx::operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp, typename _Cond >
bool **__gnu_cxx::operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
void **__gnu_cxx::swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)

5.555.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (throw_value, throw_allocator) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type forced_exception_error.

Definition in file [throw_allocator.h](#).

5.556 time_members.h File Reference

Namespaces

- namespace [std](#)

5.556.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [time_members.h](#).

5.557 `trace_fn_imps.hpp` File Reference

5.557.1 Detailed Description

Contains an implementation class for a `binary_heap`.

Definition in file [binary_heap_/trace_fn_imps.hpp](#).

5.558 `trace_fn_imps.hpp` File Reference

5.558.1 Detailed Description

Contains implementations of `cc_ht_map_`'s trace-mode functions.

Definition in file [cc_hash_table_map_/trace_fn_imps.hpp](#).

5.559 `trace_fn_imps.hpp` File Reference

5.559.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

Definition in file [gp_hash_table_map_/trace_fn_imps.hpp](#).

5.560 `trace_fn_imps.hpp` File Reference

5.560.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [left_child_next_sibling_heap_/trace_fn_imps.hpp](#).

5.561 `trace_fn_imps.hpp` File Reference

5.561.1 Detailed Description

Contains implementations of `lu_map_`.

Definition in file [list_update_map_/trace_fn_imps.hpp](#).

5.562 `trace_fn_imps.hpp` File Reference

5.562.1 Detailed Description

Contains an implementation class for `pat_trie_`.

Definition in file [pat_trie_/trace_fn_imps.hpp](#).

5.563 `trace_fn_imps.hpp` File Reference

5.563.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

Definition in file [rc_binomial_heap_/trace_fn_imps.hpp](#).

5.564 `trace_fn_imps.hpp` File Reference

5.564.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

Definition in file [thin_heap_/trace_fn_imps.hpp](#).

5.565 `traits.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::bin_search_tree_traits](#)< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
Binary search tree traits, primary template.
- struct [__gnu_pbds::detail::bin_search_tree_traits](#)< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
Specialization.

Namespaces

- namespace [__gnu_pbds](#)

5.565.1 Detailed Description

Contains an implementation for `bin_search_tree_`.

Definition in file [bin_search_tree_/traits.hpp](#).

5.566 `traits.hpp` File Reference

Classes

- struct [tree_traits](#)< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >
Tree traits class, primary template.
- struct [trie_traits](#)< Key, Data, _ATraits, Node_Update, Tag, _Alloc >
Trie traits class, primary template.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

5.566.1 Detailed Description

Contains an implementation class for tree-like classes.

Definition in file [branch_policy/traits.hpp](#).

5.567 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits](#)< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
Tree traits.
- struct [__gnu_pbds::detail::tree_traits](#)< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
Specialization.

Namespaces

- namespace [__gnu_pbds](#)

5.567.1 Detailed Description

Contains an implementation class for `ov_tree_`.

Definition in file [ov_tree_map_/traits.hpp](#).

5.568 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::trie_traits](#)< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
Specialization.
- struct [__gnu_pbds::detail::trie_traits](#)< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
Specialization.

Namespaces

- namespace [__gnu_pbds](#)

5.568.1 Detailed Description

Contains an implementation class for pat_trie_.

Definition in file [pat_trie_/traits.hpp](#).

5.569 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >](#)
Specialization.
- struct [__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >](#)
Specialization.

Namespaces

- namespace [__gnu_pbds](#)

5.569.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file [rb_tree_map_/traits.hpp](#).

5.570 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >](#)
Specialization.
- struct [__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >](#)
Specialization.

Namespaces

- namespace [__gnu_pbds](#)

5.570.1 Detailed Description

Contains an implementation for splay_tree_.

Definition in file [splay_tree_/traits.hpp](#).

5.571 tree_policy.hpp File Reference

Classes

- class [__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >](#)
Functor updating ranks of entrees.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_BRANCH_POLICY_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.571.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_policy.hpp](#).

5.572 tree_trace_base.hpp File Reference

5.572.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_trace_base.hpp](#).

5.573 trie_policy.hpp File Reference

Classes

- class [__gnu_pbds::trie_order_statistics_node_update](#)< Node_Cltr, Node_Itr, ATraits, _Alloc >
Functor updating ranks of entrees.
- class [__gnu_pbds::trie_prefix_search_node_update](#)< Node_Cltr, Node_Itr, ATraits, _Alloc >
A node updatator that allows tries to be searched for the range of values that match a certain prefix.
- struct [__gnu_pbds::trie_string_access_traits](#)< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_TRIE_POLICY_BASE`

5.573.1 Detailed Description

Contains trie-related policies.

Definition in file [trie_policy.hpp](#).

5.574 `trie_policy_base.hpp` File Reference

Classes

- class [__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >](#)
Base class for trie policies.

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.574.1 Detailed Description

Contains an implementation of `trie_policy_base`.

Definition in file [trie_policy_base.hpp](#).

5.575 `trie_string_access_traits_imp.hpp` File Reference

5.575.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

Definition in file [trie_string_access_traits_imp.hpp](#).

5.576 `tuple` File Reference

Classes

- struct [_Tuple_impl< _Idx, _Elements >](#)
- struct [std::_Tuple_impl< _Idx >](#)
- struct [std::_Tuple_impl< _Idx, _Head, _Tail...>](#)
- class [std::tuple< _Elements >](#)
Primary class template, tuple.
- class [std::tuple< _T1, _T2 >](#)
Partial specialization, 2-element tuple. Includes construction and assignment from a pair.
- struct [std::tuple_element< 0, tuple< _Head, _Tail...> >](#)
- struct [std::tuple_element< __i, tuple< _Head, _Tail...> >](#)

- struct `std::tuple_size< tuple< _Elements...> >`
class tuple_size
- struct `std::uses_allocator< tuple< _Types...>, _Alloc >`
Partial specialization for tuples.
- struct `tuple_element< __i, _Tp >`
Gives the type of the ith element of a given tuple type.
- struct `tuple_size< _Tp >`
Finds the size of a given tuple type.

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_TUPLE`

Typedefs

- template<typename _Tp >
using `std::__empty_not_final` = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp >>::type

Functions

- template<std::size_t __i, typename _Head, typename... _Tail>
constexpr `__add_ref< _Head >::type` `std::__get_helper (_Tuple_impl< __i, _Head, _Tail...> &__t)` noexcept
- template<std::size_t __i, typename _Head, typename... _Tail>
constexpr `__add_c_ref< _Head >::type` `std::__get_helper (const _Tuple_impl< __i, _Head, _Tail...> &__t)` noexcept
- template<typename... _Elements>
`tuple< _Elements &&...>` `std::forward_as_tuple (_Elements &&...__args)` noexcept
- template<std::size_t __i, typename... _Elements>
constexpr `__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >`
`std::get (tuple< _Elements...> &__t)` noexcept
- template<std::size_t __i, typename... _Elements>
constexpr `__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type >`
`std::get (const tuple< _Elements...> &__t)` noexcept
- template<std::size_t __i, typename... _Elements>
constexpr `__add_r_ref< typename tuple_element< __i, tuple< _Elements...> >::type >`
`std::get (tuple< _Elements...> &&__t)` noexcept
- template<typename... _Elements>
constexpr `tuple< typename __decay_and_strip< _Elements >::__type...>` `std::make_tuple (_Elements &&...__args)`

- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator!= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator== (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator>= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _Elements>`
`void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`
`tuple< _Elements &...> std::tie (_Elements &... __args) noexcept`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`
`constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls...>::__type`

Variables

- `const _Swallow_assign std::ignore`

5.576.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tuple](#).

5.577 type_traits File Reference

Classes

- struct [common_type< _Tp >](#)
common_type
- class [result_of< _Signature >](#)
result_of
- struct [std::add_const< _Tp >](#)
add_const
- struct [std::add_cv< _Tp >](#)
add_cv
- struct [std::add_lvalue_reference< _Tp >](#)
add_lvalue_reference
- struct [std::add_pointer< _Tp >](#)
add_pointer
- struct [std::add_rvalue_reference< _Tp >](#)
add_rvalue_reference
- struct [std::add_volatile< _Tp >](#)
add_volatile

- struct [std::aligned_storage< _Len, _Align >](#)
Alignment type.
- struct [std::alignment_of< _Tp >](#)
alignment_of
- struct [std::conditional< _Cond, _Iftrue, _Iffalse >](#)
Define a member typedef `type` to one of two argument types.
- class [std::decay< _Tp >](#)
decay
- struct [std::enable_if< bool, _Tp >](#)
Define a member typedef `type` only if a boolean constant is true.
- struct [std::extent< typename, _Uint >](#)
extent
- struct [std::has_trivial_copy_assign< _Tp >](#)
has_trivial_copy_assign (temporary legacy)
- struct [std::has_trivial_copy_constructor< _Tp >](#)
has_trivial_copy_constructor (temporary legacy)
- struct [std::has_trivial_default_constructor< _Tp >](#)
has_trivial_default_constructor (temporary legacy)
- struct [std::has_virtual_destructor< _Tp >](#)
has_virtual_destructor
- struct [std::integral_constant< _Tp, __v >](#)
integral_constant
- struct [std::is_abstract< _Tp >](#)
is_abstract
- struct [std::is_arithmetic< _Tp >](#)
is_arithmetic
- struct [std::is_array< typename >](#)
is_array
- struct [std::is_assignable< _Tp, _Up >](#)
is_assignable
- struct [std::is_base_of< _Base, _Derived >](#)
is_base_of
- struct [std::is_class< _Tp >](#)
is_class
- struct [std::is_compound< _Tp >](#)
is_compound
- struct [std::is_const< typename >](#)
is_const
- struct [std::is_constructible< _Tp, _Args >](#)
is_constructible
- struct [std::is_convertible< _From, _To >](#)
is_convertible
- struct [std::is_copy_assignable< _Tp >](#)
is_copy_assignable
- struct [std::is_copy_constructible< _Tp >](#)
is_copy_constructible
- struct [std::is_default_constructible< _Tp >](#)

- is_default_constructible*
- struct [std::is_destructible< _Tp >](#)
 - is_destructible*
- struct [std::is_empty< _Tp >](#)
 - is_empty*
- struct [std::is_enum< _Tp >](#)
 - is_enum*
- struct [std::is_floating_point< _Tp >](#)
 - is_floating_point*
- struct [std::is_function< typename >](#)
 - is_function*
- struct [std::is_fundamental< _Tp >](#)
 - is_fundamental*
- struct [std::is_integral< _Tp >](#)
 - is_integral*
- struct [std::is_literal_type< _Tp >](#)
 - is_literal_type*
- struct [std::is_lvalue_reference< typename >](#)
 - is_lvalue_reference*
- struct [std::is_member_function_pointer< _Tp >](#)
 - is_member_function_pointer*
- struct [std::is_member_object_pointer< _Tp >](#)
 - is_member_object_pointer*
- struct [std::is_member_pointer< _Tp >](#)
 - is_member_pointer*
- struct [std::is_move_assignable< _Tp >](#)
 - is_move_assignable*
- struct [std::is_move_constructible< _Tp >](#)
 - is_move_constructible*
- struct [std::is_nothrow_assignable< _Tp, _Up >](#)
 - is_nothrow_assignable*
- struct [std::is_nothrow_constructible< _Tp, _Args >](#)
 - is_nothrow_constructible*
- struct [std::is_nothrow_copy_assignable< _Tp >](#)
 - is_nothrow_copy_assignable*
- struct [std::is_nothrow_copy_constructible< _Tp >](#)
 - is_nothrow_copy_constructible*
- struct [std::is_nothrow_default_constructible< _Tp >](#)
 - is_nothrow_default_constructible*
- struct [std::is_nothrow_destructible< _Tp >](#)
 - is_nothrow_destructible*
- struct [std::is_nothrow_move_assignable< _Tp >](#)
 - is_nothrow_move_assignable*
- struct [std::is_nothrow_move_constructible< _Tp >](#)
 - is_nothrow_move_constructible*
- struct [std::is_object< _Tp >](#)
 - is_object*

- struct `std::is_pod< _Tp >`
is_pod
- struct `std::is_pointer< _Tp >`
is_pointer
- struct `std::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference
- struct `std::is_same< typename, typename >`
is_same
- struct `std::is_scalar< _Tp >`
is_scalar
- struct `std::is_signed< _Tp >`
is_signed
- struct `std::is_standard_layout< _Tp >`
is_standard_layout
- struct `std::is_trivial< _Tp >`
is_trivial
- struct `std::is_trivially_destructible< _Tp >`
is_trivially_constructible (still unimplemented)
- struct `std::is_union< _Tp >`
is_union
- struct `std::is_unsigned< _Tp >`
is_unsigned
- struct `std::is_void< _Tp >`
is_void
- struct `std::is_volatile< typename >`
is_volatile
- struct `std::make_signed< _Tp >`
make_signed
- struct `std::make_unsigned< _Tp >`
make_unsigned
- struct `std::rank< typename >`
rank
- class `std::reference_wrapper< _Tp >`
Primary class template for reference_wrapper.
- struct `std::remove_all_extents< _Tp >`
remove_all_extents
- struct `std::remove_const< _Tp >`
remove_const
- struct `std::remove_cv< _Tp >`
remove_cv
- struct `std::remove_extent< _Tp >`
remove_extent
- struct `std::remove_pointer< _Tp >`

- remove_pointer*
- struct `std::remove_reference< _Tp >`
remove_reference
- struct `std::remove_volatile< _Tp >`
remove_volatile
- struct `std::underlying_type< _Tp >`
The underlying type of an enum.

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`
- `#define _GLIBCXX_TYPE_TRAITS`

Typedefs

- `template<typename... _Cond>`
`using std::Require = typename enable_if< __and< _Cond...>::value >::type`
- `typedef integral_constant`
`< bool, false > std::false_type`
- `typedef integral_constant`
`< bool, true > std::true_type`

Functions

- `template<typename _Tp >`
`add_rvalue_reference< _Tp >::type std::declval () noexcept`

5.577.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [type_traits](#).

5.577.2 Macro Definition Documentation

5.577.2.1 `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`

Use SFINAE to determine if the type `_Tp` has a publicly-accessible member type `_NTYPE`.

Definition at line 2044 of file `type_traits`.

5.578 `type_traits` File Reference

Classes

- struct [__reflection_typelist< _Elements >](#)
- struct [std::tr2::__reflection_typelist< _First, _Rest...>](#)
Partial specialization.
- struct [std::tr2::__reflection_typelist<>](#)
Specialization for an empty typelist.
- struct [std::tr2::bases< _Tp >](#)
Sequence abstraction metafunctions for manipulating a typelist.
- struct [std::tr2::direct_bases< _Tp >](#)
Enumerate all the direct base classes of a class. Form of a typelist.

Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

Macros

- `#define _GLIBCXX_TR2_TYPE_TRAITS`

5.578.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file [tr2/type_traits](#).

5.579 `type_traits.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type)`

5.579.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [type_traits.h](#).

5.580 `type_utils.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

Typedefs

- typedef
std::tr1::integral_constant
< int, 0 > [__gnu_pbds::detail::false_type](#)
- typedef
std::tr1::integral_constant
< int, 1 > [__gnu_pbds::detail::true_type](#)

5.580.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

Definition in file [type_utils.hpp](#).

5.581 `typeindex` File Reference

Classes

- struct [std::hash< type_index >](#)
std::hash specialization for type_index.
- struct [std::type_index](#)
Class type_index
The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBCXX_TYPEINDEX`

5.581.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeindex](#).

5.582 `typeinfo` File Reference

Classes

- class [std::bad_cast](#)
*Thrown during incorrect typecasting.
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class [std::bad_typeid](#)
Thrown when a `NULL` pointer in a `typeid` expression is used.
- class [std::type_info](#)
Part of RTTI.

Namespaces

- namespace [std](#)

Macros

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`
- `#define _TYPEINFO`

5.582.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeinfo](#).

5.583 `typelist.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)
- namespace [__gnu_cxx::typelist](#)

Macros

- `#define _GLIBCXX_TPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TPELIST_CHAIN16(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)`
- `#define _GLIBCXX_TPELIST_CHAIN17(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)`

- `#define _GLIBCXX_TYPELIST_CHAIN18(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)`
- `#define _GLIBCXX_TYPELIST_CHAIN19(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN20(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

Functions

- `template<typename Fn, typename Typelist >`
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Gn, typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`

5.583.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file [typelist.h](#).

5.584 types.h File Reference

Namespaces

- namespace [__gnu_parallel](#)

Typedefs

- `typedef int64_t __gnu_parallel::_CASable`
- `typedef uint64_t __gnu_parallel::_SequenceIndex`
- `typedef uint16_t __gnu_parallel::_ThreadIndex`

Enumerations

- enum [__gnu_parallel::_AlgorithmStrategy](#) { **heuristic**, **force_sequential**, **force_parallel** }
- enum [__gnu_parallel::_FindAlgorithm](#) { **GROWING_BLOCKS**, **CONSTANT_SIZE_BLOCKS**, **EQUAL_SPLIT** }
- enum [__gnu_parallel::_MultiwayMergeAlgorithm](#) { **LOSER_TREE** }
- enum [__gnu_parallel::_Parallelism](#) {
[__gnu_parallel::sequential](#), [__gnu_parallel::parallel_unbalanced](#), [__gnu_parallel::parallel_balanced](#), [__gnu_parallel::parallel_omp_loop](#),
[__gnu_parallel::parallel_omp_loop_static](#), [__gnu_parallel::parallel_taskqueue](#) }
- enum [__gnu_parallel::_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [__gnu_parallel::_SortAlgorithm](#) { **MWMS**, **QS**, **QS_BALANCED** }
- enum [__gnu_parallel::_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

Variables

- static const int [__gnu_parallel::_CASable_bits](#)
- static const [_CASable](#) [__gnu_parallel::_CASable_mask](#)

5.584.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [types.h](#).

5.585 types_traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::no_throw_copies< Key, Mapped >](#)
Primary template.
- struct [__gnu_pbds::detail::no_throw_copies< Key, null_type >](#)
Specialization.
- struct [__gnu_pbds::detail::stored_data< _Tv, _Th >](#)
Primary template for representation of stored data. Two types of data can be stored: value and hash.
- struct [__gnu_pbds::detail::stored_data< _Tv, null_type >](#)
Specialization for representation of stored data of just value type.
- struct [__gnu_pbds::detail::stored_hash< _Th >](#)
Stored hash.
- struct [__gnu_pbds::detail::stored_value< _Tv >](#)
Stored value.
- struct [__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >](#)
- struct [__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >](#)
- struct [__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >](#)
- struct [__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >](#)
- struct [__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >](#)
Type base dispatch.
- struct [__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >](#)
Traits for abstract types.
- struct [type_base< Key, Mapped, _Alloc, Store_Hash >](#)
Primary template.

Namespaces

- namespace [__gnu_pbds](#)

5.585.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file [types_traits.hpp](#).

5.586 `unique_copy.h` File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >
_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator >
_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`

5.586.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [unique_copy.h](#).

5.587 `unique_ptr.h` File Reference

Classes

- struct [std::default_delete<_Tp>](#)
Primary template, default_delete.
- struct [std::default_delete<_Tp\[\]>](#)
Specialization, default_delete.
- struct [std::hash<unique_ptr<_Tp, _Dp>>](#)
std::hash specialization for unique_ptr.
- class [std::unique_ptr<_Tp, _Dp>](#)
20.7.1.2 unique_ptr for single objects.
- class [std::unique_ptr<_Tp\[\], _Dp>](#)
20.7.1.3 unique_ptr for array objects with a runtime length

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp >`
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`

5.587.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [unique_ptr.h](#).

5.588 unordered_map File Reference

Macros

- `#define _GLIBCXX_UNORDERED_MAP`

5.588.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_map](#).

5.589 unordered_map File Reference

Classes

- class [std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>](#)
Class std::unordered_map with safety/checking/debug instrumentation.
- class [std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>](#)
Class std::unordered_multimap with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_MAP`

Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
bool std::__debug::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const
unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
bool std::__debug::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const
unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
bool std::__debug::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const
unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
bool std::__debug::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const
unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
void std::__debug::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key,
_Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
void std::__debug::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_
multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`

5.589.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_map](#).

5.590 unordered_map File Reference

Classes

- class [std::__profile::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>](#)
Class std::unordered_map wrapper with performance instrumentation.
- class [std::__profile::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>](#)
Class std::unordered_multimap wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Macros

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_PROFILE_UNORDERED_MAP`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__profile::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__profile::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__profile::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__profile::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`
`void std::__profile::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`
`void std::__profile::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`

5.590.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_map](#).

5.591 unordered_map.h File Reference

Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Namespaces

- namespace `std`

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
using **std::__umap_hashtable** = _Hashtable<_Key, [std::pair](#)<const _Key, _Tp>, _Alloc, __detail::Select1st, _Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr>
- template<bool _Cache>
using **std::__umap_traits** = __detail::_Hashtable_traits<_Cache, false, true>
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
using **std::__ummap_hashtable** = _Hashtable<_Key, [std::pair](#)<const _Key, _Tp>, _Alloc, __detail::Select1st, _Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr>
- template<bool _Cache>
using **std::__ummap_traits** = __detail::_Hashtable_traits<_Cache, false, false>

Functions

- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
bool **std::operator!=** (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
bool **std::operator!=** (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
bool **std::operator==** (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
bool **std::operator==** (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
void **std::swap** (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
void **std::swap** (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)

5.591.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

Definition in file [unordered_map.h](#).

5.592 unordered_set File Reference

Macros

- `#define _GLIBCXX_UNORDERED_SET`

5.592.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_set](#).

5.593 unordered_set File Reference

Classes

- class [std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>](#)
Class std::unordered_multiset with safety/checking/debug instrumentation.
- class [std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>](#)
Class std::unordered_set with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_SET`

Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__debug::operator!= (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__debug::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__debug::operator== (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

5.593.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_set](#).

5.594 unordered_set File Reference

Classes

- class [std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
Unordered_multiset wrapper with performance instrumentation.
- class [std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >](#)
Unordered_set wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Macros

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_PROFILE_UNORDERED_SET`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator!= (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator== (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`
`bool std::__profile::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc>`
`void std::__profile::swap (unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`
`void std::__profile::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

5.594.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_set](#).

5.595 unordered_set.h File Reference

Classes

- class [std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class [std::unordered_set< _Value, _Hash, _Pred, _Alloc >](#)
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Namespaces

- namespace [std](#)

Typedefs

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>`
`using std::__umset_hashtable = _Hashtable< _Value, _Value, _Alloc, __detail::Identity, _Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr >`
- `template<bool _Cache>`
`using std::__umset_traits = __detail::Hashtable_traits< _Cache, true, false >`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>`
`using std::__uset_hashtable = _Hashtable< _Value, _Value, _Alloc, __detail::Identity, _Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr >`
- `template<bool _Cache>`
`using std::__uset_traits = __detail::Hashtable_traits< _Cache, true, true >`

Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc>`
`bool std::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

5.595.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

Definition in file [unordered_set.h](#).

5.596 update_fn_imps.hpp File Reference

5.596.1 Detailed Description

Contains an implementation class for `pat_trie_`.

Definition in file [update_fn_imps.hpp](#).

5.597 utility File Reference

Namespaces

- namespace [std](#)

Macros

- `#define _GLIBCXX_UTILITY`

Functions

- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`constexpr tuple_element< _Int,`
`std::pair< _Tp1, _Tp2 >`
`>::type & std::get (std::pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`constexpr tuple_element< _Int,`
`std::pair< _Tp1, _Tp2 >`
`>::type && std::get (std::pair< _Tp1, _Tp2 > &&__in) noexcept`

- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`constexpr const tuple_element`
`< _Int, std::pair< _Tp1, _Tp2 >`
`>::type & std::get (const std::pair< _Tp1, _Tp2 > &__in) noexcept`

5.597.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [utility](#).

5.598 valarray File Reference

Classes

- class [std::valarray](#)< [_Tp](#) >
Smart array designed to support numeric processing.

Namespaces

- namespace [std](#)

Macros

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`
- `#define _GLIBCXX_VALARRAY`

Functions

- `template<class _Tp >`
`_Tp * std::begin (valarray< _Tp > &__va)`
- `template<class _Tp >`
`const _Tp * std::begin (const valarray< _Tp > &__va)`
- `template<class _Tp >`
`_Tp * std::end (valarray< _Tp > &__va)`
- `template<class _Tp >`
`const _Tp * std::end (const valarray< _Tp > &__va)`
- `template<typename _Tp >`
`_Expr< _BinClos`
`< __not_equal_to, _ValArray,`
`_Constant, _Tp, _Tp >`
`, typename __fun`
`< __not_equal_to, _Tp >`
`::result_type > std::operator!= (const valarray< _Tp > &__v, const _Tp &__t)`

- template<typename _Tp >
 _Expr< _BinClos
 < __not_equal_to, _Constant,
 _ValArray, _Tp, _Tp >
 , typename __fun
 < __not_equal_to, _Tp >
 ::result_type > **std::operator!=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
 _Expr< _BinClos
 < __not_equal_to, _ValArray,
 _ValArray, _Tp, _Tp >
 , typename __fun
 < __not_equal_to, _Tp >
 ::result_type > **std::operator!=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
 _Expr< _BinClos< __modulus,
 _ValArray, _ValArray, _Tp, _Tp >
 , typename __fun< __modulus,
 _Tp >::result_type > **std::operator%** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
 _Expr< _BinClos< __modulus,
 _ValArray, _Constant, _Tp, _Tp >
 , typename __fun< __modulus,
 _Tp >::result_type > **std::operator%** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
 _Expr< _BinClos< __modulus,
 _Constant, _ValArray, _Tp, _Tp >
 , typename __fun< __modulus,
 _Tp >::result_type > **std::operator%** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
 _Expr< _BinClos< __bitwise_and,
 _ValArray, _Constant, _Tp, _Tp >
 , typename __fun
 < __bitwise_and, _Tp >
 ::result_type > **std::operator&** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
 _Expr< _BinClos< __bitwise_and,
 _Constant, _ValArray, _Tp, _Tp >
 , typename __fun
 < __bitwise_and, _Tp >
 ::result_type > **std::operator&** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
 _Expr< _BinClos< __bitwise_and,
 _ValArray, _ValArray, _Tp, _Tp >
 , typename __fun
 < __bitwise_and, _Tp >
 ::result_type > **std::operator&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
 _Expr< _BinClos< __logical_and,
 _ValArray, _Constant, _Tp, _Tp >
 , typename __fun
 < __logical_and, _Tp >
 ::result_type > **std::operator&&** (const valarray< _Tp > &__v, const _Tp &__t)

- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __logical_and, _Tp >`
`::result_type > std::operator&& (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun`
`< __logical_and, _Tp >`
`::result_type > std::operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __multiplies,`
`_Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __multiplies,`
`_Tp >::result_type > std::operator* (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __multiplies,`
`_Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __plus, _Tp >`
`::result_type > std::operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __plus, _Tp >`
`::result_type > std::operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __plus, _Tp >`
`::result_type > std::operator+ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __minus, _Tp >`
`::result_type > std::operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __minus, _Tp >`
`::result_type > std::operator- (const valarray< _Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __minus,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __minus, _Tp >`
`::result_type > std::operator- (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __divides,`
`_Tp >::result_type > std::operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __divides,`
`_Tp >::result_type > std::operator/ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __divides,`
`_Tp >::result_type > std::operator/ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __less, _Tp >`
`::result_type > std::operator< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __less, _Tp >`
`::result_type > std::operator< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __less, _Tp >`
`::result_type > std::operator< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __shift_left,`
`_Tp >::result_type > std::operator<< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __shift_left,`
`_Tp >::result_type > std::operator<< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __shift_left,`
`_Tp >::result_type > std::operator<< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __less_equal,`

```

    _Tp >::result_type > std::operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __less_equal,
    _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less_equal,
    _Tp >::result_type > std::operator<= (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __less_equal,
    _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,
    _Tp >::result_type > std::operator<= (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
  _Expr< _BinClos< __equal_to,
    _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
    _Tp >::result_type > std::operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __equal_to,
    _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __equal_to,
    _Tp >::result_type > std::operator== (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __equal_to,
    _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
    _Tp >::result_type > std::operator== (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
  _Expr< _BinClos< __greater,
    _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __greater,
    _Tp >::result_type > std::operator> (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __greater,
    _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
    _Tp >::result_type > std::operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __greater,
    _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
    _Tp >::result_type > std::operator> (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
  _Expr< _BinClos
    < __greater_equal, _ValArray,
    _ValArray, _Tp, _Tp >
  , typename __fun
    < __greater_equal, _Tp >
  ::result_type > std::operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos
    < __greater_equal, _Constant,
    _ValArray, _Tp, _Tp >
  , typename __fun
    < __greater_equal, _Tp >

```

```

::result_type > std::operator>= (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
  _Expr< _BinClos
    < __greater_equal, _ValArray,
    _Constant, _Tp, _Tp >
  , typename __fun
    < __greater_equal, _Tp >
  ::result_type > std::operator>= (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __shift_right,
    _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
    < __shift_right, _Tp >
  ::result_type > std::operator>> (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
  _Expr< _BinClos< __shift_right,
    _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
    < __shift_right, _Tp >
  ::result_type > std::operator>> (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __shift_right,
    _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
    < __shift_right, _Tp >
  ::result_type > std::operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
    _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
    < __bitwise_xor, _Tp >
  ::result_type > std::operator^ (const valarray< _Tp > &__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
    _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
    < __bitwise_xor, _Tp >
  ::result_type > std::operator^ (const _Tp &__t, const valarray< _Tp > &__v)
• template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
    _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
    < __bitwise_xor, _Tp >
  ::result_type > std::operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
    _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
    _Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
    _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __bitwise_or,
    _Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const _Tp &__t)

```

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __bitwise_or,`
`_Tp >::result_type > std::operator | (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, typename __fun< __logical_or,`
`_Tp >::result_type > std::operator || (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or,`
`_ValArray, _Constant, _Tp, _Tp >`
`, typename __fun< __logical_or,`
`_Tp >::result_type > std::operator || (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or,`
`_Constant, _ValArray, _Tp, _Tp >`
`, typename __fun< __logical_or,`
`_Tp >::result_type > std::operator || (const _Tp &__t, const valarray< _Tp > &__v)`

5.598.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [valarray](#).

5.599 valarray_after.h File Reference

Namespaces

- namespace [std](#)

Macros

- `#define _DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define _DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define _DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

Functions

- `template<class _Dom >`
`_Expr< _UnClos< _Abs, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::abs (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Abs,`
`_ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`

- `template<class _Dom >`
`_Expr< _UnClos< _Acos, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Acos,`
`_ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Asin, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Asin,`
`_ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan,`
`_ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr,`
`_Expr, _Dom1, _Dom2 >`
`, typename _Dom1::value_type > std::atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr,`
`_ValArray, _Dom, typename`
`_Dom::value_type >, typename`
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2,`
`_ValArray, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename _Dom::value_type > std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr<`
`_Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr,`
`_Constant, _Dom, typename`
`_Dom::value_type >, typename`
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`
`_Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2,`
`_Constant, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename _Dom::value_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2,`
`_ValArray, _ValArray, _Tp, _Tp >`
`, _Tp > std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2,`
`_ValArray, _Constant, _Tp, _Tp >`
`, _Tp > std::atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2,`
`_Constant, _ValArray, _Tp, _Tp >`
`, _Tp > std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cos, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::cos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cos,`
`_ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh,`
`_ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Exp, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Exp,`
`_ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log,`
`_ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr,`
`_Dom >, typename`
`_Dom::value_type > std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log10,`
`_ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos`
`< __not_equal_to, _Constant,`
`_Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun`
`< __not_equal_to, typename`
`_Dom::value_type >`
`::result_type > std::operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`
`::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos`
`< __not_equal_to, _Expr,`
`_ValArray, _Dom, typename`
`_Dom::value_type >, typename`
`__fun< __not_equal_to,`
`typename _Dom::value_type >`
`::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos`
`< __not_equal_to, _ValArray,`
`_Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun`
`< __not_equal_to, typename`
`_Dom::value_type >`
`::result_type > std::operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos`
`< __not_equal_to, _Expr, _Expr,`
`_Dom1, _Dom2 >, typename __fun`
`< __not_equal_to, typename`
`_Dom1::value_type >`
`::result_type > std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos`
`< __not_equal_to, _Expr,`
`_Constant, _Dom, typename`
`_Dom::value_type >, typename`
`__fun< __not_equal_to,`
`typename _Dom::value_type >`
`::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename`
`_Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus,`
`_Expr, _Constant, _Dom,`
`typename _Dom::value_type >`
`, typename __fun< __modulus,`
`typename _Dom::value_type >`
`::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename`
`_Dom::value_type &__t)`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __modulus,`
`_Expr, _Expr, _Dom1, _Dom2 >`
`, typename __fun< __modulus,`
`typename _Dom1::value_type >`
`::result_type > std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`

```

    _Expr< _BinClos< __modulus,
    _Constant, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __modulus,
    typename _Dom::value_type >
    ::result_type > std::operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
    ::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __modulus,
    _Expr, _ValArray, _Dom,
    typename _Dom::value_type >
    , typename __fun< __modulus,
    typename _Dom::value_type >
    ::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
    typename _Dom::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __modulus,
    _ValArray, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __modulus,
    typename _Dom::value_type >
    ::result_type > std::operator% (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
    typename _Dom::value_type > &__e)
• template<class _Dom >
    _Expr< _BinClos< __bitwise_and,
    _Expr, _Constant, _Dom,
    typename _Dom::value_type >
    , typename __fun
    < __bitwise_and, typename
    _Dom::value_type >
    ::result_type > std::operator& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename
    _Dom::value_type &__t)
• template<class _Dom >
    _Expr< _BinClos< __bitwise_and,
    _Constant, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun
    < __bitwise_and, typename
    _Dom::value_type >
    ::result_type > std::operator& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
    ::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __bitwise_and,
    _Expr, _ValArray, _Dom,
    typename _Dom::value_type >
    , typename __fun
    < __bitwise_and, typename
    _Dom::value_type >
    ::result_type > std::operator& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
    typename _Dom::value_type > &__v)
• template<class _Dom >

```

```

    _Expr< _BinClos< __bitwise_and,
    _ValArray, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun
    < __bitwise_and, typename
    _Dom::value_type >
    ::result_type > std::operator& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
    typename _Dom::value_type > &__e)
• template<class _Dom1 , class _Dom2 >
    _Expr< _BinClos< __bitwise_and,
    _Expr, _Expr, _Dom1, _Dom2 >
    , typename __fun
    < __bitwise_and, typename
    _Dom1::value_type >
    ::result_type > std::operator& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
    _Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom1 , class _Dom2 >
    _Expr< _BinClos< __logical_and,
    _Expr, _Expr, _Dom1, _Dom2 >
    , typename __fun
    < __logical_and, typename
    _Dom1::value_type >
    ::result_type > std::operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
    _Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
    _Expr< _BinClos< __logical_and,
    _Constant, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun
    < __logical_and, typename
    _Dom::value_type >
    ::result_type > std::operator&& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _-
    Dom::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __logical_and,
    _Expr, _ValArray, _Dom,
    typename _Dom::value_type >
    , typename __fun
    < __logical_and, typename
    _Dom::value_type >
    ::result_type > std::operator&& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
    typename _Dom::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __logical_and,
    _ValArray, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun
    < __logical_and, typename
    _Dom::value_type >
    ::result_type > std::operator&& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
    typename _Dom::value_type > &__e)
• template<class _Dom >

```

```

    _Expr< _BinClos< __logical_and,
    _Expr, _Constant, _Dom,
    typename _Dom::value_type >
    , typename __fun
    < __logical_and, typename
    _Dom::value_type >
    ::result_type > std::operator&& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename
    _Dom::value_type &__t)
• template<class _Dom1 , class _Dom2 >
    _Expr< _BinClos< __multiplies,
    _Expr, _Expr, _Dom1, _Dom2 >
    , typename __fun< __multiplies,
    typename _Dom1::value_type >
    ::result_type > std::operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
    _Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
    _Expr< _BinClos< __multiplies,
    _Expr, _Constant, _Dom,
    typename _Dom::value_type >
    , typename __fun< __multiplies,
    typename _Dom::value_type >
    ::result_type > std::operator* (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _
    Dom::value_type &__t)
• template<class _Dom >
    _Expr< _BinClos< __multiplies,
    _Constant, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __multiplies,
    typename _Dom::value_type >
    ::result_type > std::operator* (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
    ::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __multiplies,
    _Expr, _ValArray, _Dom,
    typename _Dom::value_type >
    , typename __fun< __multiplies,
    typename _Dom::value_type >
    ::result_type > std::operator* (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
    typename _Dom::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __multiplies,
    _ValArray, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __multiplies,
    typename _Dom::value_type >
    ::result_type > std::operator* (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
    typename _Dom::value_type > &__e)
• template<class _Dom >
    _Expr< _BinClos< __plus, _Expr,
    _ValArray, _Dom, typename
    _Dom::value_type >, typename
    __fun< __plus, typename
    _Dom::value_type >
    ::result_type > std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<

```

- ```
typename _Dom::value_type > &__v)
```
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`_fun< __plus, typename`  
`_Dom::value_type >`  
`::result_type > std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _-`  
`_Dom::value_type &__t)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __plus,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _fun< __plus,`  
`typename _Dom::value_type >`  
`::result_type > std::operator+ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
  - `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __plus, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename _fun< __plus,`  
`typename _Dom1::value_type >`  
`::result_type > std::operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __plus,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _fun< __plus,`  
`typename _Dom::value_type >`  
`::result_type > std::operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __minus,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _fun< __minus,`  
`typename _Dom::value_type >`  
`::result_type > std::operator- (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __minus,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename _fun< __minus,`  
`typename _Dom::value_type >`  
`::result_type > std::operator- (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
  - `template<class _Dom >`

```

 _Expr< _BinClos< __minus,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __minus,
 typename _Dom::value_type >
 ::result_type > std::operator- (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-
 name _Dom::value_type > &__e)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __minus,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __minus,
 typename _Dom1::value_type >
 ::result_type > std::operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _-
 Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __minus,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun< __minus,
 typename _Dom::value_type >
 ::result_type > std::operator- (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _-
 Dom::value_type &__t)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __divides,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __divides,
 typename _Dom1::value_type >
 ::result_type > std::operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _-
 Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __divides,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun< __divides,
 typename _Dom::value_type >
 ::result_type > std::operator/ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _-
 Dom::value_type &__t)
• template<class _Dom >
 _Expr< _BinClos< __divides,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __divides,
 typename _Dom::value_type >
 ::result_type > std::operator/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
 ::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __divides,
 _Expr, _ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun< __divides,
 typename _Dom::value_type >
 ::result_type > std::operator/ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-
 name _Dom::value_type > &__v)

```

- `template<class _Dom >`  
`_Expr< _BinClos< __divides,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __divides,`  
`typename _Dom::value_type >`  
`::result_type > std::operator/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename __fun< __less,`  
`typename _Dom1::value_type >`  
`::result_type > std::operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __less, typename`  
`_Dom::value_type >`  
`::result_type > std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __less,`  
`typename _Dom::value_type >`  
`::result_type > std::operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-`  
`::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr,`  
`_ValArray, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __less, typename`  
`_Dom::value_type >`  
`::result_type > std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`  
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __less,`  
`typename _Dom::value_type >`  
`::result_type > std::operator< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_left,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun< __shift_left,`  
`typename _Dom1::value_type >`  
`::result_type > std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > std::operator<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > std::operator<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _`  
`_Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > std::operator<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`  
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun< __shift_left,`  
`typename _Dom::value_type >`  
`::result_type > std::operator<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __less_equal,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun< __less_equal,`  
`typename _Dom1::value_type >`  
`::result_type > std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal,`  
`_Expr, _Constant, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __less_equal,`  
`typename _Dom::value_type >`  
`::result_type > std::operator<= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal,`  
`_Expr, _ValArray, _Dom,`  
`typename _Dom::value_type >`  
`, typename __fun< __less_equal,`  
`typename _Dom::value_type >`  
`::result_type > std::operator<= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`

- ```
typename _Dom::value_type > &__v)
```
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal,`
`_Constant, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun< __less_equal,`
`typename _Dom::value_type >`
`::result_type > std::operator<= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _-`
`_Dom::value_type > &__v)`
 - `template<class _Dom >`
`_Expr< _BinClos< __less_equal,`
`_ValArray, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun< __less_equal,`
`typename _Dom::value_type >`
`::result_type > std::operator<= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
 - `template<class _Dom >`
`_Expr< _BinClos< __equal_to,`
`_Expr, _ValArray, _Dom,`
`typename _Dom::value_type >`
`, typename __fun< __equal_to,`
`typename _Dom::value_type >`
`::result_type > std::operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`
`typename _Dom::value_type > &__v)`
 - `template<class _Dom >`
`_Expr< _BinClos< __equal_to,`
`_ValArray, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun< __equal_to,`
`typename _Dom::value_type >`
`::result_type > std::operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
 - `template<class _Dom >`
`_Expr< _BinClos< __equal_to,`
`_Constant, _Expr, typename`
`_Dom::value_type, _Dom >`
`, typename __fun< __equal_to,`
`typename _Dom::value_type >`
`::result_type > std::operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _-`
`_Dom::value_type > &__v)`
 - `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to,`
`_Expr, _Expr, _Dom1, _Dom2 >`
`, typename __fun< __equal_to,`
`typename _Dom1::value_type >`
`::result_type > std::operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
 - `template<class _Dom >`

```

    _Expr< _BinClos< __equal_to,
    _Expr, _Constant, _Dom,
    typename _Dom::value_type >
    , typename __fun< __equal_to,
    typename _Dom::value_type >
    ::result_type > std::operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename
    _Dom::value_type &__t)
• template<class _Dom >
    _Expr< _BinClos< __greater,
    _Expr, _Constant, _Dom,
    typename _Dom::value_type >
    , typename __fun< __greater,
    typename _Dom::value_type >
    ::result_type > std::operator> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename
    _Dom::value_type &__t)
• template<class _Dom >
    _Expr< _BinClos< __greater,
    _Expr, ValArray, _Dom,
    typename _Dom::value_type >
    , typename __fun< __greater,
    typename _Dom::value_type >
    ::result_type > std::operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
    typename _Dom::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos< __greater,
    _ValArray, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __greater,
    typename _Dom::value_type >
    ::result_type > std::operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
    typename _Dom::value_type > &__e)
• template<class _Dom1 , class _Dom2 >
    _Expr< _BinClos< __greater,
    _Expr, _Expr, _Dom1, _Dom2 >
    , typename __fun< __greater,
    typename _Dom1::value_type >
    ::result_type > std::operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
    _Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
    _Expr< _BinClos< __greater,
    _Constant, _Expr, typename
    _Dom::value_type, _Dom >
    , typename __fun< __greater,
    typename _Dom::value_type >
    ::result_type > std::operator> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
    ::value_type > &__v)
• template<class _Dom >
    _Expr< _BinClos
    < __greater_equal, _Expr,
    _ValArray, _Dom, typename
    _Dom::value_type >, typename
    __fun< __greater_equal,
    typename _Dom::value_type >
    ::result_type > std::operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<

```

- ```
typename _Dom::value_type > &__v)
```
- `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _Constant,`  
`_Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __greater_equal, typename`  
`_Dom::value_type >`  
`::result_type > std::operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _-`  
`_Dom::value_type > &__v)`
  - `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _ValArray,`  
`_Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename __fun`  
`< __greater_equal, typename`  
`_Dom::value_type >`  
`::result_type > std::operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
  - `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos`  
`< __greater_equal, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename __fun`  
`< __greater_equal, typename`  
`_Dom1::value_type >`  
`::result_type > std::operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
  - `template<class _Dom >`  
`_Expr< _BinClos`  
`< __greater_equal, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`__fun< __greater_equal,`  
`typename _Dom::value_type >`  
`::result_type > std::operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename`  
`_Dom::value_type &__t)`
  - `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __shift_right,`  
`_Expr, _Expr, _Dom1, _Dom2 >`  
`, typename __fun`  
`< __shift_right, typename`  
`_Dom1::value_type >`  
`::result_type > std::operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
  - `template<class _Dom >`

```

 _Expr< _BinClos< __shift_right,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun
 < __shift_right, typename
 _Dom::value_type >
 ::result_type > std::operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename
 _Dom::value_type &__t)
• template<class _Dom >
 _Expr< _BinClos< __shift_right,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun
 < __shift_right, typename
 _Dom::value_type >
 ::result_type > std::operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _-
 Dom::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __shift_right,
 _Expr, ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun
 < __shift_right, typename
 _Dom::value_type >
 ::result_type > std::operator>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
 typename _Dom::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __shift_right,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun
 < __shift_right, typename
 _Dom::value_type >
 ::result_type > std::operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
 typename _Dom::value_type > &__e)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_xor,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun
 < __bitwise_xor, typename
 _Dom::value_type >
 ::result_type > std::operator^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
 typename _Dom::value_type > &__e)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_xor,
 _Expr, ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun
 < __bitwise_xor, typename
 _Dom::value_type >
 ::result_type > std::operator^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
 typename _Dom::value_type > &__v)
• template<class _Dom1 , class _Dom2 >

```

```

 _Expr< _BinClos< __bitwise_xor,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun
 < __bitwise_xor, typename
 _Dom1::value_type >
 ::result_type > std::operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
 _Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_xor,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun
 < __bitwise_xor, typename
 _Dom::value_type >
 ::result_type > std::operator^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _-
 Dom::value_type &__t)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_xor,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun
 < __bitwise_xor, typename
 _Dom::value_type >
 ::result_type > std::operator^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
 ::value_type > &__v)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __bitwise_or,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __bitwise_or,
 typename _Dom1::value_type >
 ::result_type > std::operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _-
 Dom2, typename _Dom2::value_type > &__w)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_or,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun< __bitwise_or,
 typename _Dom::value_type >
 ::result_type > std::operator| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _-
 Dom::value_type &__t)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_or,
 _Expr, _ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun< __bitwise_or,
 typename _Dom::value_type >
 ::result_type > std::operator| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-
 name _Dom::value_type > &__v)
• template<class _Dom >

```

```

 _Expr< _BinClos< __bitwise_or,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __bitwise_or,
 typename _Dom::value_type >
 ::result_type > std::operator| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
 ::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __bitwise_or,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __bitwise_or,
 typename _Dom::value_type >
 ::result_type > std::operator| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-
 name _Dom::value_type > &__e)
• template<class _Dom >
 _Expr< _BinClos< __logical_or,
 _ValArray, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __logical_or,
 typename _Dom::value_type >
 ::result_type > std::operator|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom,
 typename _Dom::value_type > &__e)
• template<class _Dom >
 _Expr< _BinClos< __logical_or,
 _Expr, _Constant, _Dom,
 typename _Dom::value_type >
 , typename __fun< __logical_or,
 typename _Dom::value_type >
 ::result_type > std::operator|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _-
 Dom::value_type &__t)
• template<class _Dom >
 _Expr< _BinClos< __logical_or,
 _Expr, _ValArray, _Dom,
 typename _Dom::value_type >
 , typename __fun< __logical_or,
 typename _Dom::value_type >
 ::result_type > std::operator|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<
 typename _Dom::value_type > &__v)
• template<class _Dom >
 _Expr< _BinClos< __logical_or,
 _Constant, _Expr, typename
 _Dom::value_type, _Dom >
 , typename __fun< __logical_or,
 typename _Dom::value_type >
 ::result_type > std::operator|| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom-
 ::value_type > &__v)
• template<class _Dom1 , class _Dom2 >
 _Expr< _BinClos< __logical_or,
 _Expr, _Expr, _Dom1, _Dom2 >
 , typename __fun< __logical_or,
 typename _Dom1::value_type >
 ::result_type > std::operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
 _Dom2, typename _Dom2::value_type > &__w)

```

- `template<class _Dom >`  
`_Expr< _BinClos< _Pow,`  
`_ValArray, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _Dom::value_type > std::pow (const valarray< typename _Dom::valarray > &__v, const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow,`  
`_Constant, _Expr, typename`  
`_Dom::value_type, _Dom >`  
`, typename _Dom::value_type > std::pow (const typename _Dom::value_type &__t, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow,`  
`_ValArray, _Constant, _Tp, _Tp >`  
`, _Tp > std::pow (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr,`  
`_Constant, _Dom, typename`  
`_Dom::value_type >, typename`  
`_Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow,`  
`_ValArray, _ValArray, _Tp, _Tp >`  
`, _Tp > std::pow (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr,`  
`_ValArray, _Dom, typename`  
`_Dom::value_type >, typename`  
`_Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`  
`typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Pow, _Expr,`  
`_Expr, _Dom1, _Dom2 >`  
`, typename _Dom1::value_type > std::pow (const _Expr< _Dom1, typename _Dom1::value_type > &__e1,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow,`  
`_Constant, _ValArray, _Tp, _Tp >`  
`, _Tp > std::pow (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sin, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > std::sin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sin,`  
`_ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sinh, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > std::sinh (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`  
`_Expr< _UnClos< _Sinh,`  
`_ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sqrt,`  
`_ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sqrt, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tan,`  
`_ValArray, _Tp >, _Tp > std::tan (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Tan, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > std::tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Tanh, _Expr,`  
`_Dom >, typename`  
`_Dom::value_type > std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tanh,`  
`_ValArray, _Tp >, _Tp > std::tanh (const valarray< _Tp > &__v)`

#### 5.599.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray\\_after.h](#).

## 5.600 `valarray_array.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

### Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`bool > __m)`
- `template<typename _Tp >`  
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`size_t > __i)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::_Array_augmented___minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::_Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`

- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`

#### 5.600.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray\\_array.h](#).

## 5.601 valarray\_array.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define VALARRAY_ARRAY_TCC`

### Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`

#### 5.601.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray\\_array.tcc](#).

## 5.602 valarray\_before.h File Reference

### Namespaces

- namespace [std](#)

#### 5.602.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray\\_before.h](#).

## 5.603 vector File Reference

### Macros

- `#define _GLIBCXX_VECTOR`

#### 5.603.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [vector](#).

## 5.604 vector File Reference

## Classes

- class [std::\\_\\_debug::vector<\\_Tp, \\_Allocator>](#)  
*Class std::vector with safety/checking/debug instrumentation.*
- struct [std::hash<\\_\\_debug::vector<bool, \\_Alloc>>](#)  
*std::hash specialization for vector<bool>.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_VECTOR`

## Functions

- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator!= (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator< (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator<= (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator== (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator> (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator>= (const vector<_Tp, _Alloc> &__lhs, const vector<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void std::__debug::swap (vector<_Tp, _Alloc> &__lhs, vector<_Tp, _Alloc> &__rhs)`

## 5.604.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/vector](#).

## 5.605 vector File Reference

## Classes

- struct [std::hash<\\_\\_profile::vector<bool, \\_Alloc>>](#)  
*std::hash specialization for vector<bool>.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Macros

- `#define _GLIBCXX_PROFILE_VECTOR`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`

## 5.605.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/vector](#).

## 5.606 vector.tcc File Reference

## Namespaces

- namespace [std](#)

## Macros

- `#define _VECTOR_TCC`

## 5.606.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

Definition in file [vector.tcc](#).

## 5.607 vstring.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >](#)  
*Template class \_\_versa\_string.*  
*Data structure managing sequences of characters and character-like objects.*
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_u16vstring >](#)  
*std::hash specialization for \_\_u16vstring.*
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_u32vstring >](#)  
*std::hash specialization for \_\_u32vstring.*
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_vstring >](#)  
*std::hash specialization for \_\_vstring.*
- struct [std::hash< \\_\\_gnu\\_cxx::\\_\\_wvstring >](#)  
*std::hash specialization for \_\_wvstring.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> basic\\_istream< \\_CharT, \\_Traits > & std::getline \(basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is, \\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_str, \\_CharT \\_\\_delim\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> basic\\_istream< \\_CharT, \\_Traits > & std::getline \(basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is, \\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_str\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> bool \\_\\_gnu\\_cxx::operator!= \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> bool \\_\\_gnu\\_cxx::operator!= \(const \\_CharT \\*\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> bool \\_\\_gnu\\_cxx::operator!= \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_CharT \\*\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(const \\_CharT \\*\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(\\_CharT \\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)

- Generated on Fri Mar 1 2013 11:26:48 for libstdc++ by Doxygen

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator<= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >`  
`& __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_`  
`string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`  
`__enable_if< std::is_char`  
`< _CharT >::value, bool >`  
`::type __gnu_cxx::operator== (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator<`  
`_CharT >, _Base > & __lhs, const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`  
`>, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator== (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & _`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * _`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_`  
`string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * _`  
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator> (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & _`  
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_`  
`string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT`  
`* __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator>= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >`  
`& __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__`  
`versa_string< _CharT, _Traits, _Alloc, _Base > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string< _CharT, _`  
`Traits, _Alloc, _Base > & __rhs)`

#### 5.607.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [vstring.h](#).

## 5.608 vstring.tcc File Reference

## Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Macros

- `#define _VSTRING_TCC`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`

## 5.608.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file `vstring.tcc`.

5.609 `vstring_fwd.h` File Reference

## Classes

- class `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`
- class `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

#### Namespaces

- namespace `__gnu_cxx`

#### Typedefs

- typedef `__versa_string`< char, `std::char_traits`< char >, `std::allocator`< char >, `__rc_string_base` > `__gnu_cxx::__rc_string`
- typedef `vstring` `__gnu_cxx::__sso_string`
- typedef `__versa_string`< `char16_t`, `std::char_traits`< `char16_t` >, `std::allocator`< `char16_t` >, `__rc_string_base` > `__gnu_cxx::__u16rc_string`
- typedef `__u16vstring` `__gnu_cxx::__u16sso_string`
- typedef `__versa_string`< `char16_t` > `__gnu_cxx::__u16vstring`
- typedef `__versa_string`< `char32_t`, `std::char_traits`< `char32_t` >, `std::allocator`< `char32_t` >, `__rc_string_base` > `__gnu_cxx::__u32rc_string`
- typedef `__u32vstring` `__gnu_cxx::__u32sso_string`
- typedef `__versa_string`< `char32_t` > `__gnu_cxx::__u32vstring`
- typedef `__versa_string`< char > `__gnu_cxx::__vstring`
- typedef `__versa_string`< `wchar_t`, `std::char_traits`< `wchar_t` >, `std::allocator`< `wchar_t` >, `__rc_string_base` > `__gnu_cxx::__wrc_string`
- typedef `__wvstring` `__gnu_cxx::__wsso_string`
- typedef `__versa_string`< `wchar_t` > `__gnu_cxx::__wvstring`

#### 5.609.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file `vstring_fwd.h`.

## 5.610 `vstring_util.h` File Reference

#### Namespaces

- namespace `__gnu_cxx`

### 5.610.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file [vstring\\_util.h](#).

## 5.611 `workstealing.h` File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_Job<\\_DifferenceTp>](#)  
*One \_\_job for a certain thread.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define _GLIBCXX_JOB_VOLATILE`

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_workstealing (_RAIter __begin, _RAIter __end, _-  
Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter >-  
::difference_type __bound)`

### 5.611.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing. Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [workstealing.h](#).

## Index

- ~\_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase, [933](#)
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue, [958](#)
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::\_Safe\_sequence\_base, [843](#)
- ~\_Safe\_unordered\_container\_base
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, [850](#)
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_versa\_string, [654](#)
- ~auto\_ptr
  - std::auto\_ptr, [1537](#)
- ~basic\_filebuf
  - std::basic\_filebuf, [1552](#)
- ~basic\_fstream
  - std::basic\_fstream, [1579](#)
- ~basic\_ifstream
  - std::basic\_ifstream, [1626](#)
- ~basic\_ios
  - std::basic\_ios, [1664](#)
- ~basic\_iostream
  - std::basic\_iostream, [1689](#)
- ~basic\_istream
  - std::basic\_istream, [1735](#)
- ~basic\_istreamstream
  - std::basic\_istreamstream, [1776](#)
- ~basic\_ofstream
  - std::basic\_ofstream, [1814](#)
- ~basic\_ostream
  - std::basic\_ostream, [1847](#)
- ~basic\_ostringstream
  - std::basic\_ostringstream, [1880](#)
- ~basic\_regex
  - std::basic\_regex, [1907](#)
- ~basic\_streambuf
  - std::basic\_streambuf, [1916](#)
- ~basic\_string
  - std::basic\_string, [1937](#)
- ~basic\_stringstream
  - std::basic\_stringstream, [2001](#)
- ~collate
  - std::collate, [2090](#)
- ~ctype
  - std::ctype< char >, [2122](#)
  - std::ctype< wchar\_t >, [2133](#)
- ~deque
  - std::deque, [2179](#)
- ~facet
  - std::locale::facet, [2418](#)
- ~forward\_list
  - std::forward\_list, [2229](#)
- ~gslice
  - Numeric Arrays, [85](#)
- ~ios\_base
  - std::ios\_base, [2309](#)
- ~locale
  - std::locale, [2411](#)
- ~match\_results
  - std::match\_results, [2453](#)
- ~messages
  - std::messages, [2470](#)
- ~money\_get
  - std::money\_get, [2479](#)
- ~money\_put
  - std::money\_put, [2483](#)
- ~moneypunct
  - std::moneypunct, [2489](#)
- ~num\_get
  - std::num\_get, [2549](#)
- ~num\_put
  - std::num\_put, [2562](#)
- ~numpunct
  - std::numpunct, [2597](#)
- ~sentry
  - std::basic\_ostream::sentry, [1871](#)
- ~stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf, [767](#)
- ~temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer, [803](#)
- ~time\_get
  - std::time\_get, [2721](#)
- ~time\_put
  - std::time\_put, [2737](#)
- ~type\_info
  - std::type\_info, [2769](#)
- ~vector
  - std::vector, [2869](#)
- \_\_gnu\_parallel
  - parallel\_balanced, [334](#)
  - parallel\_omp\_loop, [334](#)
  - parallel\_omp\_loop\_static, [334](#)
  - parallel\_taskqueue, [334](#)
  - parallel\_unbalanced, [334](#)
  - sequential, [334](#)
- \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger
  - external\_load\_access, [994](#)
- \_\_gnu\_pbds::container\_traits
  - erase\_can\_throw, [1005](#)
  - order\_preserving, [1005](#)

- reverse\_iteration, 1006
- split\_join\_can\_throw, 1006
- \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger
  - external\_load\_access, 1181
- \_\_gnu\_pbds::lu\_counter\_policy
  - max\_count, 1192
- \_AlgorithmStrategy
  - \_\_gnu\_parallel, 334
- \_AnyMatcher
  - Base and Implementation Classes, 213
- \_AutomatonPtr
  - Base and Implementation Classes, 212
- \_BinIndex
  - \_\_gnu\_parallel, 333
- \_Bind< \_Signature >, 1259
- \_Bind\_result< \_Result, \_Signature >, 1259
- \_Bit\_scan\_forward
  - \_\_gnu\_cxx, 304
- \_CASable
  - \_\_gnu\_parallel, 333
- \_CASable\_bits
  - \_\_gnu\_parallel, 367
- \_CASable\_mask
  - \_\_gnu\_parallel, 368
- \_Construct
  - std, 507
- \_DRandomShufflingGlobalData
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 910
- \_Destroy
  - std, 507, 508
- \_Distance\_precision
  - \_\_gnu\_debug, 320
- \_FindAlgorithm
  - \_\_gnu\_parallel, 334
- \_Find\_first
  - SGI, 10
- \_Find\_next
  - SGI, 10
- \_GLIBCXX\_CALL
  - compiletime\_settings.h, 2972
- \_GLIBCXX\_MERGESORT
  - features.h, 3025
- \_GLIBCXX\_QUICKSORT
  - features.h, 3025
- \_GLIBCXX\_VOLATILE
  - partition.h, 3129
  - queue.h, 3151
- \_GuardedIterator
  - \_\_gnu\_parallel::\_GuardedIterator, 916
- \_Hash\_node< \_Value, \_Cache\_hash\_code >, 1261
- \_Hashtable\_ebo\_helper< \_Nm, \_Tp, \_\_use\_ebo >, 1261
- \_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase, 933
- \_M\_allocate\_and\_copy
  - std::\_\_detail::\_\_Nfa, 1395
  - std::vector, 2870
- \_M\_allocate\_single\_object
  - \_\_gnu\_cxx::bitmap\_allocator, 708
- \_M\_attach
  - \_\_gnu\_debug::Safe\_iterator, 817
  - \_\_gnu\_debug::Safe\_iterator\_base, 825
  - \_\_gnu\_debug::Safe\_local\_iterator, 829
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 836
  - \_\_gnu\_debug::Safe\_sequence, 840
  - \_\_gnu\_debug::Safe\_sequence\_base, 843
  - \_\_gnu\_debug::Safe\_unordered\_container, 846
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 850
  - \_\_gnu\_debug::basic\_string, 857
  - std::\_\_debug::deque, 1307
  - std::\_\_debug::forward\_list, 1311
  - std::\_\_debug::list, 1317
  - std::\_\_debug::map, 1321
  - std::\_\_debug::multimap, 1326
  - std::\_\_debug::multiset, 1331
  - std::\_\_debug::set, 1336
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1345
  - std::\_\_debug::unordered\_multiset, 1350
  - std::\_\_debug::unordered\_set, 1355
  - std::\_\_debug::vector, 1361
- \_M\_attach\_local
  - \_\_gnu\_debug::Safe\_unordered\_container, 846
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 850
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1345
  - std::\_\_debug::unordered\_multiset, 1350
  - std::\_\_debug::unordered\_set, 1355
- \_M\_attach\_local\_single
  - \_\_gnu\_debug::Safe\_unordered\_container, 846
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 850
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1346
  - std::\_\_debug::unordered\_multiset, 1350
  - std::\_\_debug::unordered\_set, 1355
- \_M\_attach\_single
  - \_\_gnu\_debug::Safe\_iterator, 817, 818
  - \_\_gnu\_debug::Safe\_iterator\_base, 825
  - \_\_gnu\_debug::Safe\_local\_iterator, 830
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 836
  - \_\_gnu\_debug::Safe\_sequence, 840
  - \_\_gnu\_debug::Safe\_sequence\_base, 843
  - \_\_gnu\_debug::Safe\_unordered\_container, 846
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 850
  - \_\_gnu\_debug::basic\_string, 857

- std::\_\_debug::deque, 1307
- std::\_\_debug::forward\_list, 1311
- std::\_\_debug::list, 1317
- std::\_\_debug::map, 1321
- std::\_\_debug::multimap, 1326
- std::\_\_debug::multiset, 1331
- std::\_\_debug::set, 1336
- std::\_\_debug::unordered\_map, 1341
- std::\_\_debug::unordered\_multimap, 1346
- std::\_\_debug::unordered\_multiset, 1351
- std::\_\_debug::unordered\_set, 1356
- std::\_\_debug::vector, 1361
- \_M\_attached\_to
  - \_\_gnu\_debug::\_Safe\_iterator, 818
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 825
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 830
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 837
- \_M\_before\_dereferenceable
  - \_\_gnu\_debug::\_Safe\_iterator, 818
- \_M\_begin
  - \_\_gnu\_parallel::\_Piece, 948
- \_M\_bin\_proc
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 910
- \_M\_bins\_begin
  - \_\_gnu\_parallel::\_DRSSorterPU, 912
- \_M\_buf
  - \_\_gnu\_cxx::enc\_filebuf, 728
  - \_\_gnu\_cxx::stdio\_filebuf, 781
  - std::basic\_filebuf, 1566
- \_M\_buf\_locale
  - \_\_gnu\_cxx::enc\_filebuf, 729
  - \_\_gnu\_cxx::stdio\_filebuf, 781
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 799
  - std::basic\_filebuf, 1566
  - std::basic\_streambuf, 1929
  - std::basic\_stringbuf, 1990
- \_M\_buf\_size
  - \_\_gnu\_cxx::enc\_filebuf, 729
  - \_\_gnu\_cxx::stdio\_filebuf, 781
  - std::basic\_filebuf, 1566
- \_M\_can\_compare
  - \_\_gnu\_debug::\_Safe\_iterator, 818
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 825
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 830
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 837
- \_M\_clear
  - \_\_gnu\_cxx::free\_list, 735
- \_M\_comp
  - \_\_gnu\_parallel::\_LoserTree, 927
  - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 931
  - \_\_gnu\_parallel::\_LoserTreeBase, 934
- \_M\_const\_iterators
  - \_\_gnu\_debug::\_Safe\_sequence, 841
- \_\_gnu\_debug::\_Safe\_sequence\_base, 844
- \_\_gnu\_debug::\_Safe\_unordered\_container, 848
- \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 851
- \_\_gnu\_debug::basic\_string, 873
- std::\_\_debug::deque, 1308
- std::\_\_debug::forward\_list, 1313
- std::\_\_debug::list, 1318
- std::\_\_debug::map, 1323
- std::\_\_debug::multimap, 1328
- std::\_\_debug::multiset, 1333
- std::\_\_debug::set, 1338
- std::\_\_debug::unordered\_map, 1342
- std::\_\_debug::unordered\_multimap, 1347
- std::\_\_debug::unordered\_multiset, 1352
- std::\_\_debug::unordered\_set, 1357
- std::\_\_debug::vector, 1362
- \_M\_const\_local\_iterators
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 848
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 851
- std::\_\_debug::unordered\_map, 1342
- std::\_\_debug::unordered\_multimap, 1347
- std::\_\_debug::unordered\_multiset, 1352
- std::\_\_debug::unordered\_set, 1357
- \_M\_create\_node
  - std::list, 2396
- \_M\_create\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 716
  - \_\_gnu\_cxx::stdio\_filebuf, 767
  - std::basic\_filebuf, 1552
- \_M\_data
  - std::\_List\_node, 1466
- \_M\_deallocate\_single\_object
  - \_\_gnu\_cxx::bitmap\_allocator, 708
- \_M\_dereferenceable
  - \_\_gnu\_debug::\_Safe\_iterator, 818
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 830
- \_M\_destroy\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 716
  - \_\_gnu\_cxx::stdio\_filebuf, 767
  - std::basic\_filebuf, 1552
- \_M\_detach
  - \_\_gnu\_debug::\_Safe\_iterator, 818
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 825
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 830
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 837
  - \_\_gnu\_debug::\_Safe\_sequence, 840
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 843
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 846
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 850
  - \_\_gnu\_debug::basic\_string, 857
  - std::\_\_debug::deque, 1307

- std::\_\_debug::forward\_list, 1312
- std::\_\_debug::list, 1317
- std::\_\_debug::map, 1322
- std::\_\_debug::multimap, 1327
- std::\_\_debug::multiset, 1331
- std::\_\_debug::set, 1336
- std::\_\_debug::unordered\_map, 1341
- std::\_\_debug::unordered\_multimap, 1346
- std::\_\_debug::unordered\_multiset, 1351
- std::\_\_debug::unordered\_set, 1356
- std::\_\_debug::vector, 1361
- \_M\_detach\_all
  - \_\_gnu\_debug::\_Safe\_sequence, 840
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 843
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 846
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 850
  - \_\_gnu\_debug::basic\_string, 857
  - std::\_\_debug::deque, 1307
  - std::\_\_debug::forward\_list, 1312
  - std::\_\_debug::list, 1317
  - std::\_\_debug::map, 1322
  - std::\_\_debug::multimap, 1327
  - std::\_\_debug::multiset, 1332
  - std::\_\_debug::set, 1337
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1346
  - std::\_\_debug::unordered\_multiset, 1351
  - std::\_\_debug::unordered\_set, 1356
  - std::\_\_debug::vector, 1361
- \_M\_detach\_local
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 846
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 850
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1346
  - std::\_\_debug::unordered\_multiset, 1351
  - std::\_\_debug::unordered\_set, 1356
- \_M\_detach\_local\_single
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 846
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 850
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1346
  - std::\_\_debug::unordered\_multiset, 1351
  - std::\_\_debug::unordered\_set, 1356
- \_M\_detach\_single
  - \_\_gnu\_debug::\_Safe\_iterator, 818
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 825
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 830
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 837
  - \_\_gnu\_debug::\_Safe\_sequence, 840
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 843
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 846
- \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 851
- \_\_gnu\_debug::basic\_string, 857
- std::\_\_debug::deque, 1307
- std::\_\_debug::forward\_list, 1312
- std::\_\_debug::list, 1317
- std::\_\_debug::map, 1322
- std::\_\_debug::multimap, 1327
- std::\_\_debug::multiset, 1332
- std::\_\_debug::set, 1337
- std::\_\_debug::unordered\_map, 1341
- std::\_\_debug::unordered\_multimap, 1346
- std::\_\_debug::unordered\_multiset, 1351
- std::\_\_debug::unordered\_set, 1356
- std::\_\_debug::vector, 1361
- \_M\_detach\_singular
  - \_\_gnu\_debug::\_Safe\_sequence, 840
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 843
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 847
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 851
  - \_\_gnu\_debug::basic\_string, 857
  - std::\_\_debug::deque, 1307
  - std::\_\_debug::forward\_list, 1312
  - std::\_\_debug::list, 1317
  - std::\_\_debug::map, 1322
  - std::\_\_debug::multimap, 1327
  - std::\_\_debug::multiset, 1332
  - std::\_\_debug::set, 1337
  - std::\_\_debug::unordered\_map, 1341
  - std::\_\_debug::unordered\_multimap, 1346
  - std::\_\_debug::unordered\_multiset, 1351
  - std::\_\_debug::unordered\_set, 1356
  - std::\_\_debug::vector, 1361
- \_M\_dist
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 910
- \_M\_elements\_leftover
  - \_\_gnu\_parallel::\_QSBThreadLocal, 954
- \_M\_end
  - \_\_gnu\_parallel::\_Piece, 948
- \_M\_ext\_buf
  - \_\_gnu\_cxx::enc\_filebuf, 729
  - \_\_gnu\_cxx::stdio\_filebuf, 781
  - std::basic\_filebuf, 1567
- \_M\_ext\_buf\_size
  - \_\_gnu\_cxx::enc\_filebuf, 729
  - \_\_gnu\_cxx::stdio\_filebuf, 781
  - std::basic\_filebuf, 1567
- \_M\_ext\_next
  - \_\_gnu\_cxx::enc\_filebuf, 729
  - \_\_gnu\_cxx::stdio\_filebuf, 782
  - std::basic\_filebuf, 1567
- \_M\_fill\_initialize
  - std::deque, 2179

- `_M_finish_iterator`
  - `__gnu_parallel::__accumulate_selector`, 875
  - `__gnu_parallel::__adjacent_difference_selector`, 876
  - `__gnu_parallel::__count_if_selector`, 882
  - `__gnu_parallel::__count_selector`, 884
  - `__gnu_parallel::__fill_selector`, 885
  - `__gnu_parallel::__for_each_selector`, 889
  - `__gnu_parallel::__generate_selector`, 891
  - `__gnu_parallel::__generic_for_each_selector`, 893
  - `__gnu_parallel::__identity_selector`, 894
  - `__gnu_parallel::__inner_product_selector`, 896
  - `__gnu_parallel::__replace_if_selector`, 903
  - `__gnu_parallel::__replace_selector`, 905
  - `__gnu_parallel::__transform1_selector`, 906
  - `__gnu_parallel::__transform2_selector`, 908
- `_M_first`
  - `__gnu_parallel::_Job`, 921
- `_M_first_insert`
  - `__gnu_parallel::_LoserTree`, 928
  - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 931
  - `__gnu_parallel::_LoserTreeBase`, 934
- `_M_gcount`
  - `std::basic_fstream`, 1613
  - `std::basic_ifstream`, 1653
  - `std::basic_iostream`, 1722
  - `std::basic_istream`, 1761
  - `std::basic_istreamstream`, 1802
  - `std::basic_stringstream`, 2035
- `_M_get`
  - `__gnu_cxx::free_list`, 735
- `_M_get_mutex`
  - `__gnu_debug::_Safe_iterator`, 818
  - `__gnu_debug::_Safe_iterator_base`, 825
  - `__gnu_debug::_Safe_local_iterator`, 830
  - `__gnu_debug::_Safe_local_iterator_base`, 837
  - `__gnu_debug::_Safe_sequence`, 840
  - `__gnu_debug::_Safe_sequence_base`, 844
  - `__gnu_debug::_Safe_unordered_container`, 847
  - `__gnu_debug::_Safe_unordered_container_base`, 851
  - `__gnu_debug::basic_string`, 857
  - `std::__debug::deque`, 1307
  - `std::__debug::forward_list`, 1312
  - `std::__debug::list`, 1317
  - `std::__debug::map`, 1322
  - `std::__debug::multimap`, 1327
  - `std::__debug::multiset`, 1332
  - `std::__debug::set`, 1337
  - `std::__debug::unordered_map`, 1341
  - `std::__debug::unordered_multimap`, 1346
  - `std::__debug::unordered_multiset`, 1351
  - `std::__debug::unordered_set`, 1356
  - `std::__debug::vector`, 1361
- `_M_getloc`
  - `std::basic_fstream`, 1579
  - `std::basic_ifstream`, 1626
  - `std::basic_ios`, 1665
  - `std::basic_iostream`, 1689
  - `std::basic_istream`, 1735
  - `std::basic_istreamstream`, 1776
  - `std::basic_ofstream`, 1814
  - `std::basic_ostream`, 1847
  - `std::basic_ostreamstream`, 1880
  - `std::basic_stringstream`, 2001
  - `std::ios_base`, 2309
- `_M_global`
  - `__gnu_parallel::_QSBThreadLocal`, 955
- `_M_in_beg`
  - `__gnu_cxx::enc_filebuf`, 729
  - `__gnu_cxx::stdio_filebuf`, 782
  - `__gnu_cxx::stdio_sync_filebuf`, 799
  - `std::basic_filebuf`, 1567
  - `std::basic_streambuf`, 1929
  - `std::basic_stringbuf`, 1991
- `_M_in_cur`
  - `__gnu_cxx::enc_filebuf`, 729
  - `__gnu_cxx::stdio_filebuf`, 782
  - `__gnu_cxx::stdio_sync_filebuf`, 799
  - `std::basic_filebuf`, 1567
  - `std::basic_streambuf`, 1929
  - `std::basic_stringbuf`, 1991
- `_M_in_end`
  - `__gnu_cxx::enc_filebuf`, 729
  - `__gnu_cxx::stdio_filebuf`, 782
  - `__gnu_cxx::stdio_sync_filebuf`, 799
  - `std::basic_filebuf`, 1567
  - `std::basic_streambuf`, 1929
  - `std::basic_stringbuf`, 1991
- `_M_in_same_bucket`
  - `__gnu_debug::_Safe_local_iterator`, 831
- `_M_incrementable`
  - `__gnu_debug::_Safe_iterator`, 819
  - `__gnu_debug::_Safe_local_iterator`, 831
- `_M_initial`
  - `__gnu_parallel::_QSBThreadLocal`, 955
- `_M_initialize_map`
  - `std::_Deque_base`, 1448
  - `std::deque`, 2180
- `_M_insert`
  - `__gnu_cxx::free_list`, 735
- `_M_invalidate`
  - `__gnu_debug::_Safe_iterator`, 819
  - `__gnu_debug::_Safe_iterator_base`, 825
  - `__gnu_debug::_Safe_local_iterator`, 831
  - `__gnu_debug::_Safe_local_iterator_base`, 837
- `_M_invalidate_all`
  - `__gnu_debug::_Safe_sequence`, 840

- \_\_gnu\_debug::Safe\_sequence\_base, 844
- \_\_gnu\_debug::Safe\_unordered\_container, 847
- \_\_gnu\_debug::Safe\_unordered\_container\_base, 851
- \_\_gnu\_debug::basic\_string, 858
- std::\_\_debug::deque, 1307
- std::\_\_debug::forward\_list, 1312
- std::\_\_debug::list, 1317
- std::\_\_debug::map, 1322
- std::\_\_debug::multimap, 1327
- std::\_\_debug::multiset, 1332
- std::\_\_debug::set, 1337
- std::\_\_debug::unordered\_map, 1341
- std::\_\_debug::unordered\_multimap, 1346
- std::\_\_debug::unordered\_multiset, 1351
- std::\_\_debug::unordered\_set, 1356
- std::\_\_debug::vector, 1361
- \_M\_invalidate\_if
  - \_\_gnu\_debug::Safe\_sequence, 841
  - \_\_gnu\_debug::Safe\_unordered\_container, 847
  - \_\_gnu\_debug::basic\_string, 858
  - std::\_\_debug::deque, 1307
  - std::\_\_debug::forward\_list, 1312
  - std::\_\_debug::list, 1317
  - std::\_\_debug::map, 1322
  - std::\_\_debug::multimap, 1327
  - std::\_\_debug::multiset, 1332
  - std::\_\_debug::set, 1337
  - std::\_\_debug::unordered\_map, 1342
  - std::\_\_debug::unordered\_multimap, 1346
  - std::\_\_debug::unordered\_multiset, 1351
  - std::\_\_debug::unordered\_set, 1356
  - std::\_\_debug::vector, 1362
- \_M\_invalidate\_local\_if
  - \_\_gnu\_debug::Safe\_unordered\_container, 847
  - std::\_\_debug::unordered\_map, 1342
  - std::\_\_debug::unordered\_multimap, 1347
  - std::\_\_debug::unordered\_multiset, 1352
  - std::\_\_debug::unordered\_set, 1357
- \_M\_is\_before\_begin
  - \_\_gnu\_debug::Safe\_iterator, 819
- \_M\_is\_begin
  - \_\_gnu\_debug::Safe\_iterator, 819
  - \_\_gnu\_debug::Safe\_local\_iterator, 831
- \_M\_is\_binnest
  - \_\_gnu\_debug::Safe\_iterator, 819
- \_M\_is\_end
  - \_\_gnu\_debug::Safe\_iterator, 819
  - \_\_gnu\_debug::Safe\_local\_iterator, 831
- \_M\_iterators
  - \_\_gnu\_debug::Safe\_sequence, 841
  - \_\_gnu\_debug::Safe\_sequence\_base, 844
  - \_\_gnu\_debug::Safe\_unordered\_container, 848
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 852
  - \_\_gnu\_debug::basic\_string, 873
  - std::\_\_debug::deque, 1308
  - std::\_\_debug::forward\_list, 1313
  - std::\_\_debug::list, 1318
  - std::\_\_debug::map, 1323
  - std::\_\_debug::multimap, 1328
  - std::\_\_debug::multiset, 1333
  - std::\_\_debug::set, 1338
  - std::\_\_debug::unordered\_map, 1342
  - std::\_\_debug::unordered\_multimap, 1347
  - std::\_\_debug::unordered\_multiset, 1352
  - std::\_\_debug::unordered\_set, 1357
  - std::\_\_debug::vector, 1362
- \_M\_key
  - \_\_gnu\_parallel::LoserTreeBase::\_Loser, 935
- \_M\_last
  - \_\_gnu\_parallel::Job, 921
- \_M\_leftover\_parts
  - \_\_gnu\_parallel::QSBThreadLocal, 955
- \_M\_load
  - \_\_gnu\_parallel::Job, 921
- \_M\_local\_iterators
  - \_\_gnu\_debug::Safe\_unordered\_container, 848
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 852
  - std::\_\_debug::unordered\_map, 1343
  - std::\_\_debug::unordered\_multimap, 1347
  - std::\_\_debug::unordered\_multiset, 1352
  - std::\_\_debug::unordered\_set, 1357
- \_M\_log\_k
  - \_\_gnu\_parallel::LoserTree, 928
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 931
  - \_\_gnu\_parallel::LoserTreeBase, 934
- \_M\_losers
  - \_\_gnu\_parallel::LoserTree, 928
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 931
  - \_\_gnu\_parallel::LoserTreeBase, 934
- \_M\_mode
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 782
  - std::basic\_filebuf, 1567
  - std::basic\_stringbuf, 1991
- \_M\_new\_elements\_at\_back
  - std::deque, 2180
- \_M\_new\_elements\_at\_front
  - std::deque, 2180
- \_M\_next
  - \_\_gnu\_debug::Safe\_iterator, 822
  - \_\_gnu\_debug::Safe\_iterator\_base, 826
  - \_\_gnu\_debug::Safe\_local\_iterator, 833

- \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 838
- \_M\_num\_bins
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 910
- \_M\_num\_bits
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 910
- \_M\_num\_threads
  - \_\_gnu\_parallel::\_DRSSorterPU, 912
  - \_\_gnu\_parallel::\_PMWMSSortingData, 950
  - \_\_gnu\_parallel::\_QSBThreadLocal, 955
- \_M\_offsets
  - \_\_gnu\_parallel::\_PMWMSSortingData, 951
- \_M\_out\_beg
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 782
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 800
  - std::basic\_filebuf, 1568
  - std::basic\_streambuf, 1930
  - std::basic\_stringbuf, 1991
- \_M\_out\_cur
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 783
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 800
  - std::basic\_filebuf, 1568
  - std::basic\_streambuf, 1930
  - std::basic\_stringbuf, 1991
- \_M\_out\_end
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 783
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 800
  - std::basic\_filebuf, 1568
  - std::basic\_streambuf, 1930
  - std::basic\_stringbuf, 1991
- \_M\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 783
  - std::basic\_filebuf, 1568
- \_M\_pback\_cur\_save
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 783
  - std::basic\_filebuf, 1568
- \_M\_pback\_end\_save
  - \_\_gnu\_cxx::enc\_filebuf, 730
  - \_\_gnu\_cxx::stdio\_filebuf, 783
  - std::basic\_filebuf, 1569
- \_M\_pback\_init
  - \_\_gnu\_cxx::enc\_filebuf, 731
  - \_\_gnu\_cxx::stdio\_filebuf, 783
  - std::basic\_filebuf, 1569
- \_M\_pieces
  - \_\_gnu\_parallel::\_PMWMSSortingData, 951
- \_M\_pop\_back\_aux
  - std::deque, 2180
- \_M\_pop\_front\_aux
  - std::deque, 2181
- \_M\_prior
  - \_\_gnu\_debug::\_Safe\_iterator, 822
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 826
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 833
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 838
- \_M\_push\_back\_aux
  - std::deque, 2181
- \_M\_push\_front\_aux
  - std::deque, 2181
- \_M\_range\_check
  - std::\_\_detail::\_Nfa, 1395
  - std::deque, 2181
  - std::vector, 2870
- \_M\_range\_initialize
  - std::deque, 2181
- \_M\_reading
  - \_\_gnu\_cxx::enc\_filebuf, 731
  - \_\_gnu\_cxx::stdio\_filebuf, 784
  - std::basic\_filebuf, 1569
- \_M\_reallocate\_map
  - std::deque, 2182
- \_M\_reserve\_elements\_at\_back
  - std::deque, 2182
- \_M\_reserve\_elements\_at\_front
  - std::deque, 2182
- \_M\_reserve\_map\_at\_back
  - std::deque, 2182
- \_M\_reserve\_map\_at\_front
  - std::deque, 2183
- \_M\_reset
  - \_\_gnu\_debug::\_Safe\_iterator, 820
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 825
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 831
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 837
- \_M\_revalidate\_singular
  - \_\_gnu\_debug::\_Safe\_sequence, 841
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 844
  - \_\_gnu\_debug::\_Safe\_unordered\_container, 847
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 851
  - \_\_gnu\_debug::basic\_string, 858
  - std::\_\_debug::deque, 1308
  - std::\_\_debug::forward\_list, 1312
  - std::\_\_debug::list, 1317
  - std::\_\_debug::map, 1322
  - std::\_\_debug::multimap, 1327
  - std::\_\_debug::multiset, 1332
  - std::\_\_debug::set, 1337
  - std::\_\_debug::unordered\_map, 1342
  - std::\_\_debug::unordered\_multimap, 1347
  - std::\_\_debug::unordered\_multiset, 1352
  - std::\_\_debug::unordered\_set, 1357
  - std::\_\_debug::vector, 1362
- \_M\_samples

- \_\_gnu\_parallel::\_PMWMSSortingData, 951
- \_M\_sd
  - \_\_gnu\_parallel::\_DRSSorterPU, 912
- \_M\_seed
  - \_\_gnu\_parallel::\_DRSSorterPU, 912
- \_M\_sequence
  - \_\_gnu\_debug::Safe\_iterator, 822
  - \_\_gnu\_debug::Safe\_iterator\_base, 826
  - \_\_gnu\_debug::Safe\_local\_iterator, 834
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 838
- \_M\_sequential\_algorithm
  - \_\_gnu\_parallel::\_adjacent\_find\_selector, 877
  - \_\_gnu\_parallel::\_find\_first\_of\_selector, 886
  - \_\_gnu\_parallel::\_find\_if\_selector, 887
  - \_\_gnu\_parallel::\_mismatch\_selector, 898
- \_M\_set\_buffer
  - \_\_gnu\_cxx::enc\_filebuf, 716
  - \_\_gnu\_cxx::stdio\_filebuf, 767
  - std::basic\_filebuf, 1552
- \_M\_set\_node
  - std::Deque\_iterator, 1450
- \_M\_singular
  - \_\_gnu\_debug::Safe\_iterator, 820
  - \_\_gnu\_debug::Safe\_iterator\_base, 825
  - \_\_gnu\_debug::Safe\_local\_iterator, 831
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 837
- \_M\_source
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 910
  - \_\_gnu\_parallel::\_LoserTreeBase::\_Loser, 935
  - \_\_gnu\_parallel::\_PMWMSSortingData, 951
- \_M\_starts
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 911
  - \_\_gnu\_parallel::\_PMWMSSortingData, 951
- \_M\_sup
  - \_\_gnu\_parallel::\_LoserTreeBase::\_Loser, 935
- \_M\_swap
  - \_\_gnu\_debug::Safe\_sequence, 841
  - \_\_gnu\_debug::Safe\_sequence\_base, 844
  - \_\_gnu\_debug::Safe\_unordered\_container, 847
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 851
  - \_\_gnu\_debug::basic\_string, 858
  - std::\_\_debug::deque, 1308
  - std::\_\_debug::forward\_list, 1312
  - std::\_\_debug::list, 1317
  - std::\_\_debug::map, 1322
  - std::\_\_debug::multimap, 1327
  - std::\_\_debug::multiset, 1332
  - std::\_\_debug::set, 1337
  - std::\_\_debug::unordered\_map, 1342
  - std::\_\_debug::unordered\_multimap, 1347
  - std::\_\_debug::unordered\_multiset, 1352
  - std::\_\_debug::unordered\_set, 1357
  - std::\_\_debug::vector, 1362
- \_M\_temporaries
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 911
- \_M\_temporary
  - \_\_gnu\_parallel::\_PMWMSSortingData, 951
- \_M\_transfer\_from\_if
  - \_\_gnu\_debug::Safe\_sequence, 841
  - \_\_gnu\_debug::basic\_string, 858
  - std::\_\_debug::deque, 1308
  - std::\_\_debug::forward\_list, 1312
  - std::\_\_debug::list, 1318
  - std::\_\_debug::map, 1322
  - std::\_\_debug::multimap, 1327
  - std::\_\_debug::multiset, 1332
  - std::\_\_debug::set, 1337
  - std::\_\_debug::vector, 1362
- \_M\_unlink
  - \_\_gnu\_debug::Safe\_iterator, 820
  - \_\_gnu\_debug::Safe\_iterator\_base, 826
  - \_\_gnu\_debug::Safe\_local\_iterator, 832
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 837
- \_M\_use\_pointer
  - \_\_gnu\_parallel::\_LoserTreeTraits, 942
- \_M\_version
  - \_\_gnu\_debug::Safe\_iterator, 823
  - \_\_gnu\_debug::Safe\_iterator\_base, 826
  - \_\_gnu\_debug::Safe\_local\_iterator, 834
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 838
  - \_\_gnu\_debug::Safe\_sequence, 841
  - \_\_gnu\_debug::Safe\_sequence\_base, 844
  - \_\_gnu\_debug::Safe\_unordered\_container, 848
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 852
  - \_\_gnu\_debug::basic\_string, 873
  - std::\_\_debug::deque, 1308
  - std::\_\_debug::forward\_list, 1313
  - std::\_\_debug::list, 1318
  - std::\_\_debug::map, 1323
  - std::\_\_debug::multimap, 1328
  - std::\_\_debug::multiset, 1333
  - std::\_\_debug::set, 1338
  - std::\_\_debug::unordered\_map, 1343
  - std::\_\_debug::unordered\_multimap, 1347
  - std::\_\_debug::unordered\_multiset, 1352
  - std::\_\_debug::unordered\_set, 1357
  - std::\_\_debug::vector, 1362
- \_M\_w
  - std::\_Base\_bitset, 1444
  - std::tr2::dynamic\_bitset\_base, 2744
- \_M\_write
  - std::basic\_fstream, 1579
  - std::basic\_iostream, 1690
  - std::basic\_ofstream, 1815
  - std::basic\_ostream, 1847
  - std::basic\_ostringstream, 1880

- std::basic\_stringstream, 2002
- \_Matcher
  - Base and Implementation Classes, 212
- \_Mu< \_Arg, \_IsBindExp, \_IsPlaceholder >, 1263
- \_MultiwayMergeAlgorithm
  - \_\_gnu\_parallel, 334
- \_Opcode
  - Base and Implementation Classes, 213
- \_Parallelism
  - \_\_gnu\_parallel, 334
- \_PartialSumAlgorithm
  - \_\_gnu\_parallel, 334
- \_Piece
  - \_\_gnu\_parallel::QSBThreadLocal, 954
- \_PseudoSequence
  - \_\_gnu\_parallel::PseudoSequence, 952
- \_QSBThreadLocal
  - \_\_gnu\_parallel::QSBThreadLocal, 954
- \_RandomNumber
  - \_\_gnu\_parallel::RandomNumber, 956
- \_Reference\_wrapper\_base\_impl< \_Unary, \_Binary, \_Tp >, 1263
- \_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue, 957
- \_S\_invalid\_state\_id
  - Base and Implementation Classes, 213
- \_Safe\_iterator
  - \_\_gnu\_debug::Safe\_iterator, 816, 817
- \_Safe\_iterator\_base
  - \_\_gnu\_debug::Safe\_iterator\_base, 824
- \_Safe\_local\_iterator
  - \_\_gnu\_debug::Safe\_local\_iterator, 829
- \_Safe\_local\_iterator\_base
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 836
- \_SequenceIndex
  - \_\_gnu\_parallel, 333
- \_SortAlgorithm
  - \_\_gnu\_parallel, 334
- \_SplittingAlgorithm
  - \_\_gnu\_parallel, 334
- \_StatIdT
  - Base and Implementation Classes, 213
- \_StateSet
  - Base and Implementation Classes, 213
- \_StateStack
  - Base and Implementation Classes, 213
- \_Tagger
  - Base and Implementation Classes, 213
- \_Temporary\_buffer
  - std::Temporary\_buffer, 1480
- \_ThreadIndex
  - \_\_gnu\_parallel, 333
- \_TokenT
  - std::\_\_detail::Scanner, 1413
- \_Tuple\_impl< \_Idx, \_Elements >, 1264
- \_Unchecked\_flip
  - SGI, 11
- \_Unchecked\_reset
  - SGI, 11
- \_Unchecked\_set
  - SGI, 11
- \_Unchecked\_test
  - SGI, 11
- \_\_addressof
  - Utilities, 67
- \_\_allocator\_base
  - Allocators, 162
- \_\_base
  - \_\_gnu\_debug, 320
- \_\_begin1\_iterator
  - \_\_gnu\_parallel::inner\_product\_selector, 896
- \_\_begin2\_iterator
  - \_\_gnu\_parallel::inner\_product\_selector, 896
- \_\_bins\_end
  - \_\_gnu\_parallel::DRSSorterPU, 911
- \_\_bit\_allocate
  - \_\_gnu\_cxx::\_\_detail, 313
- \_\_bit\_free
  - \_\_gnu\_cxx::\_\_detail, 313
- \_\_calc\_borders
  - \_\_gnu\_parallel, 334
- \_\_check\_dereferenceable
  - \_\_gnu\_debug, 320, 321
- \_\_check\_singular
  - \_\_gnu\_debug, 321
- \_\_check\_singular\_aux
  - \_\_gnu\_debug, 321
- \_\_check\_string
  - \_\_gnu\_debug, 321
- \_\_compare\_and\_swap
  - \_\_gnu\_parallel, 335
- \_\_ctype\_type
  - std::basic\_ios, 1661
- \_\_cxxabiv1::\_\_forced\_unwind, 627
- \_\_decode2
  - \_\_gnu\_parallel, 335
- \_\_delete\_min\_insert
  - \_\_gnu\_parallel::LoserTree, 927
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 930
- \_\_determine\_samples
  - \_\_gnu\_parallel, 335
- \_\_encode2
  - \_\_gnu\_parallel, 336
- \_\_env\_t
  - \_\_gnu\_profile, 375
- \_\_equally\_split

- \_\_gnu\_parallel, 336
- \_\_equally\_split\_point
  - \_\_gnu\_parallel, 337
- \_\_fetch\_and\_add
  - \_\_gnu\_parallel, 337
- \_\_final\_insertion\_sort
  - std, 498
- \_\_find
  - std, 498
- \_\_find\_if
  - std, 498, 499
- \_\_find\_if\_not
  - std, 499
- \_\_find\_if\_not\_n
  - std, 499
- \_\_find\_template
  - \_\_gnu\_parallel, 337d
- \_\_for\_each\_template\_random\_access
  - \_\_gnu\_parallel, 339
- \_\_for\_each\_template\_random\_access\_ed
  - \_\_gnu\_parallel, 340
- \_\_for\_each\_template\_random\_access\_omp\_loop
  - \_\_gnu\_parallel, 340
- \_\_for\_each\_template\_random\_access\_workstealing
  - \_\_gnu\_parallel, 341
- \_\_gcd
  - std, 499
- \_\_genrand\_bits
  - \_\_gnu\_parallel::RandomNumber, 956
- \_\_get\_distance
  - \_\_gnu\_debug, 321
- \_\_get\_min\_source
  - \_\_gnu\_parallel::LoserTree, 927
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 930
  - \_\_gnu\_parallel::LoserTreeBase, 933
- \_\_get\_num\_threads
  - \_\_gnu\_parallel::balanced\_quicksort\_tag, 967
  - \_\_gnu\_parallel::balanced\_tag, 968
  - \_\_gnu\_parallel::default\_parallel\_tag, 970
  - \_\_gnu\_parallel::exact\_tag, 972
  - \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 975
  - \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 976
  - \_\_gnu\_parallel::multiway\_mergesort\_tag, 977
  - \_\_gnu\_parallel::omp\_loop\_static\_tag, 978
  - \_\_gnu\_parallel::omp\_loop\_tag, 980
  - \_\_gnu\_parallel::parallel\_tag, 982
  - \_\_gnu\_parallel::quicksort\_tag, 983
  - \_\_gnu\_parallel::sampling\_tag, 984
  - \_\_gnu\_parallel::unbalanced\_tag, 986
- \_\_glibcxx\_check\_erase
  - macros.h, 3089
- \_\_glibcxx\_check\_erase\_after
  - macros.h, 3089
- \_\_glibcxx\_check\_erase\_range
  - macros.h, 3090
- \_\_glibcxx\_check\_erase\_range\_after
  - macros.h, 3090
- \_\_glibcxx\_check\_heap\_pred
  - macros.h, 3090
- \_\_glibcxx\_check\_insert
  - macros.h, 3090
- \_\_glibcxx\_check\_insert\_after
  - macros.h, 3090
- \_\_glibcxx\_check\_insert\_range
  - macros.h, 3090
- \_\_glibcxx\_check\_insert\_range\_after
  - macros.h, 3090
- \_\_glibcxx\_check\_partitioned\_lower
  - macros.h, 3091
- \_\_glibcxx\_check\_partitioned\_lower\_pred
  - macros.h, 3091
- \_\_glibcxx\_check\_partitioned\_upper\_pred
  - macros.h, 3091
- \_\_glibcxx\_check\_sorted\_pred
  - macros.h, 3091
- \_\_gnu\_cxx, 287
  - \_Bit\_scan\_forward, 304
  - \_\_static\_pointer\_cast, 303
  - operator<, 306, 307
  - operator<=, 307, 308
  - operator>, 310
  - operator>=, 311
  - operator+, 305, 306
  - operator==, 308, 309
  - swap, 312
- \_\_gnu\_cxx::Caster< \_ToType >, 695
- \_\_gnu\_cxx::Char\_types< \_CharT >, 696
- \_\_gnu\_cxx::ExtPtr\_allocator< \_Tp >, 696
- \_\_gnu\_cxx::Invalid\_type, 697
- \_\_gnu\_cxx::Pointer\_adapter< \_Storage\_policy >, 698
- \_\_gnu\_cxx::Relative\_pointer\_impl< \_Tp >, 700
- \_\_gnu\_cxx::Relative\_pointer\_impl< const \_Tp >, 700
- \_\_gnu\_cxx::Std\_pointer\_impl< \_Tp >, 701
- \_\_gnu\_cxx::Unqualified\_type< \_Tp >, 701
- \_\_gnu\_cxx::\_\_alloc\_traits
  - allocate, 630
  - const\_void\_pointer, 629
  - construct, 631
  - deallocate, 631
  - destroy, 631
  - max\_size, 632
  - propagate\_on\_container\_swap, 630
  - void\_pointer, 630
- \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc >, 628
- \_\_gnu\_cxx::\_\_detail, 312
- \_\_bit\_allocate, 313

- `__bit_free`, 313
- `__num_bitmaps`, 313
- `__num_blocks`, 313
- `gnu_cxx::__detail::__Bitmap_counter< _Tp >`, 634
- `gnu_cxx::__detail::__Ffit_finder`
  - `argument_type`, 635
  - `result_type`, 636
- `gnu_cxx::__detail::__Ffit_finder< _Tp >`, 635
- `gnu_cxx::__mt_alloc< _Tp, _Poolp >`, 636
- `gnu_cxx::__mt_alloc_base< _Tp >`, 637
- `gnu_cxx::__pool< false >`, 639
- `gnu_cxx::__pool< true >`, 640
- `gnu_cxx::__pool_alloc< _Tp >`, 641
- `gnu_cxx::__pool_alloc_base`, 643
- `gnu_cxx::__pool_base`, 644
- `gnu_cxx::__scoped_lock`, 647
- `gnu_cxx::__versa_string`
  - `~__versa_string`, 654
  - `__versa_string`, 651d
- `append`, 654d
- `assign`, 656d
- `at`, 659, 660
- `back`, 660
- `begin`, 661
- `c_str`, 661
- `capacity`, 661
- `cbegin`, 661
- `cend`, 661
- `clear`, 662
- `compare`, 662d
- `copy`, 664
- `crbegin`, 665
- `crend`, 665
- `data`, 665
- `empty`, 665
- `end`, 666
- `erase`, 666, 667
- `find`, 667, 668
- `find_first_not_of`, 669, 670
- `find_first_of`, 671, 672
- `find_last_not_of`, 672, 673
- `find_last_of`, 674, 675
- `front`, 675, 676
- `get_allocator`, 676
- `insert`, 676d
- `length`, 680
- `max_size`, 680
- `npos`, 695
- `operator+=`, 680d
- `operator=`, 682, 683
- `pop_back`, 684
- `push_back`, 684
- `rbegin`, 685
- `rend`, 685
- `replace`, 685d
- `reserve`, 691
- `resize`, 691, 692
- `rfind`, 692, 693
- `shrink_to_fit`, 694
- `size`, 694
- `substr`, 694
- `swap`, 695
- `gnu_cxx::annotate_base`, 702
- `gnu_cxx::array_allocator< _Tp, _Array >`, 703
- `gnu_cxx::array_allocator_base< _Tp >`, 704
- `gnu_cxx::binary_compose`
  - `argument_type`, 706
  - `result_type`, 706
- `gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`, 705
- `gnu_cxx::bitmap_allocator`
  - `_M_allocate_single_object`, 708
  - `_M_deallocate_single_object`, 708
- `gnu_cxx::bitmap_allocator< _Tp >`, 706
- `gnu_cxx::char_traits< _CharT >`, 709
- `gnu_cxx::character< V, I, S >`, 710
- `gnu_cxx::condition_base`, 711
- `gnu_cxx::constant_unary_fun< _Result, _Argument >`, 712
- `gnu_cxx::constant_void_fun< _Result >`, 712
- `gnu_cxx::debug_allocator< _Alloc >`, 713
- `gnu_cxx::enc_filebuf`
  - `_M_buf`, 728
  - `_M_buf_locale`, 729
  - `_M_buf_size`, 729
  - `_M_create_pback`, 716
  - `_M_destroy_pback`, 716
  - `_M_ext_buf`, 729
  - `_M_ext_buf_size`, 729
  - `_M_ext_next`, 729
  - `_M_in_beg`, 729
  - `_M_in_cur`, 729
  - `_M_in_end`, 729
  - `_M_mode`, 730
  - `_M_out_beg`, 730
  - `_M_out_cur`, 730
  - `_M_out_end`, 730
  - `_M_pback`, 730
  - `_M_pback_cur_save`, 730
  - `_M_pback_end_save`, 730
  - `_M_pback_init`, 731
  - `_M_reading`, 731
  - `_M_set_buffer`, 716
- `close`, 717
- `eback`, 717
- `egptr`, 717
- `epptr`, 717
- `gbump`, 718

- getloc, 718
- gptr, 718
- imbue, 718
- in\_avail, 719
- is\_open, 719
- open, 719, 720
- overflow, 720
- pbackfail, 720
- pbase, 721
- pbump, 721
- pptr, 721
- pubimbue, 722
- pubseekoff, 722
- pubseekpos, 722
- pubsetbuf, 722
- pubsync, 723
- sbumpc, 723
- seekoff, 723
- seekpos, 723
- setbuf, 723
- setg, 724
- setp, 724
- sgetc, 724
- sgetn, 725
- showmanyc, 725
- snextc, 725
- sputbackc, 725
- sputc, 726
- sputn, 726
- sungetc, 726
- sync, 727
- uflow, 727
- underflow, 727
- xsggetn, 728
- xsgputn, 728
- \_\_gnu\_cxx::enc\_filebuf<\_CharT>, 714
- \_\_gnu\_cxx::encoding\_char\_traits<\_CharT>, 731
- \_\_gnu\_cxx::encoding\_state, 732
- \_\_gnu\_cxx::forced\_error, 733
  - what, 734
- \_\_gnu\_cxx::free\_list, 734
  - \_M\_clear, 735
  - \_M\_get, 735
  - \_M\_insert, 735
- \_\_gnu\_cxx::limit\_condition, 742
- \_\_gnu\_cxx::limit\_condition::always\_adjustor, 743
- \_\_gnu\_cxx::limit\_condition::limit\_adjustor, 743
- \_\_gnu\_cxx::limit\_condition::never\_adjustor, 743
- \_\_gnu\_cxx::malloc\_allocator<\_Tp>, 744
- \_\_gnu\_cxx::new\_allocator<\_Tp>, 745
- \_\_gnu\_cxx::project1st
  - first\_argument\_type, 746
  - result\_type, 746
  - second\_argument\_type, 747
- \_\_gnu\_cxx::project1st<\_Arg1, \_Arg2>, 746
- \_\_gnu\_cxx::project2nd
  - first\_argument\_type, 747
  - result\_type, 747
  - second\_argument\_type, 747
- \_\_gnu\_cxx::project2nd<\_Arg1, \_Arg2>, 747
- \_\_gnu\_cxx::random\_condition, 748
- \_\_gnu\_cxx::random\_condition::always\_adjustor, 749
- \_\_gnu\_cxx::random\_condition::group\_adjustor, 749
- \_\_gnu\_cxx::random\_condition::never\_adjustor, 749
- \_\_gnu\_cxx::recursive\_init\_error, 753
  - what, 753
- \_\_gnu\_cxx::rope<\_CharT, \_Alloc>, 753
- \_\_gnu\_cxx::select1st
  - argument\_type, 759
  - result\_type, 759
- \_\_gnu\_cxx::select1st<\_Pair>, 759
- \_\_gnu\_cxx::select2nd
  - argument\_type, 760
  - result\_type, 760
- \_\_gnu\_cxx::select2nd<\_Pair>, 760
- \_\_gnu\_cxx::slist<\_Tp, \_Alloc>, 760
- \_\_gnu\_cxx::stdio\_filebuf
  - ~stdio\_filebuf, 767
  - \_M\_buf, 781
  - \_M\_buf\_locale, 781
  - \_M\_buf\_size, 781
  - \_M\_create\_pback, 767
  - \_M\_destroy\_pback, 767
  - \_M\_ext\_buf, 781
  - \_M\_ext\_buf\_size, 781
  - \_M\_ext\_next, 782
  - \_M\_in\_beg, 782
  - \_M\_in\_cur, 782
  - \_M\_in\_end, 782
  - \_M\_mode, 782
  - \_M\_out\_beg, 782
  - \_M\_out\_cur, 783
  - \_M\_out\_end, 783
  - \_M\_pback, 783
  - \_M\_pback\_cur\_save, 783
  - \_M\_pback\_end\_save, 783
  - \_M\_pback\_init, 783
  - \_M\_reading, 784
  - \_M\_set\_buffer, 767
- close, 767
- eback, 767
- egptr, 768
- epptr, 768
- fd, 768
- file, 769
- gbump, 769
- getloc, 769
- gptr, 769

- imbue, 770
- in\_avail, 770
- is\_open, 770
- open, 770, 771
- overflow, 771
- pbackfail, 772
- pbase, 772
- pbump, 773
- pptr, 773
- pubimbue, 773
- pubseekoff, 773
- pubseekpos, 774
- pubsetbuf, 774
- pubsync, 774
- sbumpc, 774
- seekoff, 775
- seekpos, 775
- setbuf, 775
- setg, 776
- setp, 776
- sgetc, 776
- sgetn, 777
- showmanyc, 777
- snextc, 777
- sputbackc, 778
- sputc, 778
- sputn, 778
- stdio\_filebuf, 766
- sungetc, 779
- sync, 779
- uflow, 779
- underflow, 779
- xsggetn, 780
- xspn, 780
- \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 763
- \_\_gnu\_cxx::stdio\_sync\_filebuf
  - \_M\_buf\_locale, 799
  - \_M\_in\_beg, 799
  - \_M\_in\_cur, 799
  - \_M\_in\_end, 799
  - \_M\_out\_beg, 800
  - \_M\_out\_cur, 800
  - \_M\_out\_end, 800
  - \_\_streambuf\_type, 787
- eback, 787
- egptr, 787
- epptr, 787
- file, 788
- gbump, 788
- getloc, 788
- gptr, 788
- imbue, 789
- in\_avail, 789
- overflow, 789
- pbackfail, 790
- pbase, 790
- pbump, 791
- pptr, 791
- pubimbue, 791
- pubseekoff, 792
- pubseekpos, 792
- pubsetbuf, 792
- pubsync, 792
- sbumpc, 792
- seekoff, 793
- seekpos, 793
- setbuf, 793
- setg, 794
- setp, 794
- sgetc, 794
- sgetn, 795
- showmanyc, 795
- snextc, 795
- sputbackc, 796
- sputc, 796
- sputn, 796
- sungetc, 797
- sync, 797
- uflow, 797
- underflow, 797
- xsggetn, 798
- xspn, 798
- \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 784
- \_\_gnu\_cxx::subtractive\_rng, 800
  - argument\_type, 801
  - operator(), 801
  - result\_type, 801
  - subtractive\_rng, 801
- \_\_gnu\_cxx::temporary\_buffer
  - ~temporary\_buffer, 803
  - begin, 803
  - end, 803
  - requested\_size, 803
  - size, 804
  - temporary\_buffer, 803
- \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 802
- \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >, 804
- \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >, 805
- \_\_gnu\_cxx::throw\_allocator\_random< \_Tp >, 806
- \_\_gnu\_cxx::throw\_value\_base< \_Cond >, 808
- \_\_gnu\_cxx::throw\_value\_limit, 809
- \_\_gnu\_cxx::throw\_value\_random, 810
- \_\_gnu\_cxx::typelist, 313
  - apply\_generator, 314
- \_\_gnu\_cxx::unary\_compose
  - argument\_type, 812
  - result\_type, 812

---

- \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2  
    >, 811
- \_\_gnu\_debug, 314
  - \_Distance\_precision, 320
  - \_base, 320
  - \_check\_dereferenceable, 320, 321
  - \_check\_singular, 321
  - \_check\_singular\_aux, 321
  - \_check\_string, 321
  - \_get\_distance, 321
  - \_valid\_range, 322
  - \_valid\_range\_aux, 322
  - \_valid\_range\_aux2, 322
- \_\_gnu\_debug:: \_After\_nth\_from< \_Iterator >, 812
- \_\_gnu\_debug:: \_BeforeBeginHelper< \_Sequence >, 813
- \_\_gnu\_debug:: \_Equal\_to< \_Type >, 813
- \_\_gnu\_debug:: \_Not\_equal\_to< \_Type >, 814
- \_\_gnu\_debug:: \_Safe\_iterator
  - \_M\_attach, 817
  - \_M\_attach\_single, 817, 818
  - \_M\_attached\_to, 818
  - \_M\_before\_dereferenceable, 818
  - \_M\_can\_compare, 818
  - \_M\_dereferenceable, 818
  - \_M\_detach, 818
  - \_M\_detach\_single, 818
  - \_M\_get\_mutex, 818
  - \_M\_incrementable, 819
  - \_M\_invalidate, 819
  - \_M\_is\_begin, 819
  - \_M\_is\_beginnest, 819
  - \_M\_is\_end, 819
  - \_M\_next, 822
  - \_M\_prior, 822
  - \_M\_reset, 820
  - \_M\_sequence, 822
  - \_M\_singular, 820
  - \_M\_unlink, 820
  - \_M\_version, 823
  - \_Safe\_iterator, 816, 817
  - base, 820
  - operator \_Iterator, 820
  - operator\*, 820
  - operator++, 821
  - operator->, 821
  - operator--, 821
  - operator=, 822
- \_\_gnu\_debug:: \_Safe\_iterator< \_Iterator, \_Sequence >, 814
- \_\_gnu\_debug:: \_Safe\_iterator\_base, 823
  - \_M\_attach, 825
  - \_M\_detach, 825
  - \_M\_invalidate, 825
  - \_M\_next, 826
  - \_M\_prior, 826
  - \_M\_reset, 825
  - \_M\_sequence, 826
  - \_M\_singular, 825
  - \_M\_unlink, 826
  - \_M\_version, 826
- \_\_gnu\_debug:: \_Safe\_local\_iterator
  - \_M\_attach, 829
  - \_M\_dereferenceable, 830
  - \_M\_detach, 830
  - \_M\_incrementable, 831
  - \_M\_invalidate, 831
  - \_M\_next, 833
  - \_M\_prior, 833
  - \_M\_reset, 831
  - \_M\_sequence, 834
  - \_M\_singular, 831
  - \_M\_unlink, 832
  - \_M\_version, 834
  - base, 832
  - bucket, 832
  - operator \_Iterator, 832
  - operator\*, 832
  - operator++, 832, 833
  - operator->, 833
  - operator=, 833
- \_\_gnu\_debug:: \_Safe\_local\_iterator\_base, 835
- \_\_gnu\_debug:: \_Safe\_sequence
  - \_M\_attach, 840
  - \_M\_attach\_single, 840
  - \_M\_const\_iterators, 841
  - \_M\_detach, 840
  - \_M\_detach\_all, 840
  - \_M\_detach\_single, 840
  - \_M\_detach\_singular, 840
  - \_M\_get\_mutex, 840
  - \_M\_invalidate\_all, 840
  - \_M\_invalidate\_if, 841
  - \_M\_iterators, 841
  - \_M\_revalidate\_singular, 841
  - \_M\_swap, 841
  - \_M\_version, 841
- \_\_gnu\_debug:: \_Safe\_sequence< \_Sequence >, 839
- \_\_gnu\_debug:: \_Safe\_sequence\_base, 842
  - \_M\_attach, 843
  - \_M\_detach, 843
  - \_M\_iterators, 844
  - \_M\_swap, 844
  - \_M\_version, 844
- \_\_gnu\_debug:: \_Safe\_unordered\_container
  - \_M\_attach, 846
  - \_M\_detach, 846
  - \_M\_iterators, 848
  - \_M\_swap, 847

---

- `_M_version`, 848
- `__gnu_debug::Safe_unordered_container< _Container`  
    `>`, 845
- `__gnu_debug::Safe_unordered_container_base`, 849
- `__gnu_debug::basic_string`
  - `_M_attach`, 857
  - `_M_attach_single`, 857
  - `_M_const_iterators`, 873
  - `_M_detach`, 857
  - `_M_detach_all`, 857
  - `_M_detach_single`, 857
  - `_M_detach_singular`, 857
  - `_M_get_mutex`, 857
  - `_M_invalidate_all`, 858
  - `_M_invalidate_if`, 858
  - `_M_iterators`, 873
  - `_M_revalidate_singular`, 858
  - `_M_swap`, 858
  - `_M_transfer_from_if`, 858
  - `_M_version`, 873
- `append`, 858, 859
- `assign`, 859, 860
- `at`, 860
- `back`, 861
- `capacity`, 861
- `compare`, 861, 862
- `empty`, 862
- `erase`, 862, 863
- `find`, 863
- `find_first_not_of`, 863
- `find_first_of`, 864
- `find_last_not_of`, 864
- `find_last_of`, 864
- `front`, 865
- `get_allocator`, 865
- `insert`, 865d
- `length`, 867
- `max_size`, 868
- `npos`, 873
- `operator+=`, 868
- `replace`, 868d
- `reserve`, 872
- `rfind`, 872
- `size`, 872
- `swap`, 872
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator`  
    `>`, 852
- `__gnu_internal`, 323
- `__gnu_parallel`, 323
  - `_AlgorithmStrategy`, 334
  - `_BinIndex`, 333
  - `_CASable`, 333
  - `_CASable_bits`, 367
  - `_CASable_mask`, 368
  - `_FindAlgorithm`, 334
  - `_MultiwayMergeAlgorithm`, 334
  - `_Parallelism`, 334
  - `_PartialSumAlgorithm`, 334
  - `_SequenceIndex`, 333
  - `_SortAlgorithm`, 334
  - `_SplittingAlgorithm`, 334
  - `_ThreadIndex`, 333
  - `__calc_borders`, 334
  - `__compare_and_swap`, 335
  - `__decode2`, 335
  - `__determine_samples`, 335
  - `__encode2`, 336
  - `__equally_split`, 336
  - `__equally_split_point`, 337
  - `__fetch_and_add`, 337
  - `__find_template`, 337d
  - `__for_each_template_random_access`, 339
  - `__for_each_template_random_access_ed`, 340
  - `__is_sorted`, 342
  - `__median_of_three_iterators`, 342
  - `__merge_advance`, 342
  - `__merge_advance_movc`, 343
  - `__merge_advance_usual`, 343
  - `__parallel_merge_advance`, 344
  - `__parallel_nth_element`, 345
  - `__parallel_partial_sort`, 345
  - `__parallel_partial_sum`, 345
  - `__parallel_partial_sum_basecase`, 346
  - `__parallel_partial_sum_linear`, 346
  - `__parallel_partition`, 347
  - `__parallel_random_shuffle`, 347
  - `__parallel_random_shuffle_drs`, 347
  - `__parallel_random_shuffle_drs_pu`, 348
  - `__parallel_sort`, 348d
  - `__parallel_sort_qs`, 352
  - `__parallel_sort_qs_conquer`, 353
  - `__parallel_sort_qs_divide`, 353
  - `__parallel_sort_qsb`, 353
  - `__parallel_unique_copy`, 354
  - `__qsb_conquer`, 355
  - `__qsb_divide`, 355
  - `__qsb_local_sort_with_helping`, 355
  - `__random_number_pow2`, 356
  - `__rd_log2`, 356
  - `__round_up_to_pow2`, 356
  - `__search_template`, 356
  - `__sequential_multiway_merge`, 357
  - `__sequential_random_shuffle`, 357
  - `__shrink`, 358
  - `__shrink_and_double`, 358
  - `__yield`, 358
- `list_partition`, 358
- `max`, 359

- min, [359](#)
- multiseq\_partition, [359](#)
- multiseq\_selection, [360](#)
- multiway\_merge, [360](#)
- multiway\_merge\_3\_variant, [361](#)
- multiway\_merge\_4\_variant, [362](#)
- multiway\_merge\_exact\_splitting, [363](#)
- multiway\_merge\_loser\_tree, [363](#)
- multiway\_merge\_loser\_tree\_sentinel, [363](#)
- multiway\_merge\_loser\_tree\_unguarded, [364](#)
- multiway\_merge\_sampling\_splitting, [364](#)
- multiway\_merge\_sentinels, [365](#)
- parallel\_multiway\_merge, [366](#)
- parallel\_sort\_mwms, [367](#)
- parallel\_sort\_mwms\_pu, [367](#)
- \_\_gnu\_parallel::DRSSorterPU
  - \_M\_sd, [912](#)
  - \_M\_seed, [912](#)
- \_\_gnu\_parallel::DRandomShufflingGlobalData
  - \_M\_dist, [910](#)
  - \_M\_source, [910](#)
  - \_M\_starts, [911](#)
  - \_M\_temporaries, [911](#)
- \_\_gnu\_parallel::DRandomShufflingGlobalData< \_RAIter
  - >, [909](#)
- \_\_gnu\_parallel::DummyReduct, [912](#)
- \_\_gnu\_parallel::EqualFromLess
  - first\_argument\_type, [913](#)
  - result\_type, [914](#)
  - second\_argument\_type, [914](#)
- \_\_gnu\_parallel::EqualTo
  - first\_argument\_type, [915](#)
  - result\_type, [915](#)
  - second\_argument\_type, [915](#)
- \_\_gnu\_parallel::EqualTo< \_T1, \_T2 >, [914](#)
- \_\_gnu\_parallel::GuardedIterator
  - \_GuardedIterator, [916](#)
  - operator \_RAIter, [916](#)
  - operator<, [917](#)
  - operator<=, [917](#)
  - operator\*, [916](#)
  - operator++, [916](#)
- \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, [915](#)
- \_\_gnu\_parallel::IteratorPair
  - first, [919](#)
  - second, [919](#)
  - second\_type, [919](#)
- \_\_gnu\_parallel::Job
  - \_M\_first, [921](#)
  - \_M\_last, [921](#)
  - \_M\_load, [921](#)
- \_\_gnu\_parallel::Job< \_DifferenceTp >, [920](#)
- \_\_gnu\_parallel::\_Less
  - first\_argument\_type, [922](#)
  - result\_type, [922](#)
  - second\_argument\_type, [923](#)
- \_\_gnu\_parallel::\_Less< \_T1, \_T2 >, [922](#)
- \_\_gnu\_parallel::\_Lexicographic
  - first\_argument\_type, [924](#)
  - result\_type, [924](#)
  - second\_argument\_type, [924](#)
- \_\_gnu\_parallel::\_Lexicographic< \_T1, \_T2, \_Compare >, [923](#)
- \_\_gnu\_parallel::\_LexicographicReverse
  - first\_argument\_type, [925](#)
  - result\_type, [925](#)
  - second\_argument\_type, [926](#)
- \_\_gnu\_parallel::\_LoserTree
  - \_M\_comp, [927](#)
  - \_M\_first\_insert, [928](#)
  - \_M\_log\_k, [928](#)
  - \_M\_losers, [928](#)
  - \_delete\_min\_insert, [927](#)
  - \_get\_min\_source, [927](#)
  - \_insert\_start, [927](#)
- \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, [929](#)
- \_\_gnu\_parallel::\_LoserTreeBase
  - ~\_LoserTreeBase, [933](#)
  - \_LoserTreeBase, [933](#)
  - \_M\_comp, [934](#)
  - \_M\_first\_insert, [934](#)
  - \_M\_log\_k, [934](#)
  - \_M\_losers, [934](#)
  - \_get\_min\_source, [933](#)
  - \_insert\_start, [933](#)
- \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >, [932](#)
- \_\_gnu\_parallel::\_LoserTreePointer< false, \_Tp, \_Compare >, [937](#)
- \_\_gnu\_parallel::\_LoserTreePointerBase< \_Tp, \_Compare >, [938](#)
- \_\_gnu\_parallel::\_LoserTreePointerUnguarded< false, \_Tp, \_Compare >, [940](#)
- \_\_gnu\_parallel::\_LoserTreeTraits
  - \_M\_use\_pointer, [942](#)
- \_\_gnu\_parallel::\_LoserTreeTraits< \_Tp >, [942](#)
- \_\_gnu\_parallel::\_LoserTreeUnguarded< false, \_Tp, \_Compare >, [944](#)
- \_\_gnu\_parallel::\_LoserTreeUnguardedBase< \_Tp, \_Compare >, [945](#)
- \_\_gnu\_parallel::\_Multiplies
  - first\_argument\_type, [947](#)
  - result\_type, [947](#)
  - second\_argument\_type, [947](#)
- \_\_gnu\_parallel::\_Multiplies< \_Tp1, \_Tp2, \_Result >, [946](#)
- \_\_gnu\_parallel::\_Nothing, [947](#)
- operator(), [947](#)

- \_\_gnu\_parallel:: Piece
  - \_M\_begin, 948
  - \_M\_end, 948
- \_\_gnu\_parallel:: Piece< \_DifferenceTp >, 948
- \_\_gnu\_parallel:: Plus
  - first\_argument\_type, 949
  - result\_type, 949
  - second\_argument\_type, 950
- \_\_gnu\_parallel:: Plus< \_Tp1, \_Tp2, \_Result >, 949
- \_\_gnu\_parallel:: PseudoSequence
  - \_PseudoSequence, 952
  - begin, 952
  - end, 952
- \_\_gnu\_parallel:: PseudoSequence< \_Tp, \_DifferenceTp >, 951
- \_\_gnu\_parallel:: PseudoSequenceIterator< \_Tp, \_DifferenceTp >, 953
- \_\_gnu\_parallel:: QSBThreadLocal
  - \_M\_global, 955
  - \_M\_initial, 955
  - \_Piece, 954
- \_\_gnu\_parallel:: QSBThreadLocal< \_RAIter >, 953
- \_\_gnu\_parallel:: RandomNumber, 955
  - \_RandomNumber, 956
  - \_genrand\_bits, 956
  - operator(), 956
- \_\_gnu\_parallel:: RestrictedBoundedConcurrentQueue
  - pop\_back, 958
  - pop\_front, 958
  - push\_front, 958
- \_\_gnu\_parallel:: RestrictedBoundedConcurrentQueue< \_Tp >, 957
- \_\_gnu\_parallel:: Settings, 959
  - accumulate\_minimal\_n, 961
  - adjacent\_difference\_minimal\_n, 961
  - cache\_line\_size, 961
  - count\_minimal\_n, 961
  - fill\_minimal\_n, 961
  - find\_increasing\_factor, 961
  - find\_initial\_block\_size, 961
  - find\_maximum\_block\_size, 962
  - find\_scale\_factor, 962
  - find\_sequential\_search\_size, 962
  - for\_each\_minimal\_n, 962
  - generate\_minimal\_n, 962
  - get, 961
  - L1\_cache\_size, 962
  - L2\_cache\_size, 962
  - max\_element\_minimal\_n, 962
  - merge\_minimal\_n, 962
  - merge\_oversampling, 963
  - min\_element\_minimal\_n, 963
  - multiway\_merge\_minimal\_k, 963
  - multiway\_merge\_minimal\_n, 963
  - multiway\_merge\_oversampling, 963
  - nth\_element\_minimal\_n, 963
  - partial\_sort\_minimal\_n, 963
  - partial\_sum\_dilation, 963
  - partial\_sum\_minimal\_n, 963
  - partition\_chunk\_share, 964
  - partition\_chunk\_size, 964
  - partition\_minimal\_n, 964
  - qsb\_steals, 964
  - random\_shuffle\_minimal\_n, 964
  - replace\_minimal\_n, 964
  - search\_minimal\_n, 964
  - set, 961
  - set\_difference\_minimal\_n, 964
  - set\_intersection\_minimal\_n, 964
  - set\_symmetric\_difference\_minimal\_n, 964
  - set\_union\_minimal\_n, 965
  - sort\_minimal\_n, 965
  - sort\_mwms\_oversampling, 965
  - sort\_qs\_num\_samples\_preset, 965
  - sort\_qsb\_base\_case\_maximal\_n, 965
  - TLB\_size, 965
  - transform\_minimal\_n, 965
  - unique\_copy\_minimal\_n, 965
- \_\_gnu\_parallel:: \_\_accumulate\_binop\_reduct< \_BinOp >, 873
- \_\_gnu\_parallel:: \_\_accumulate\_selector
  - operator(), 875
- \_\_gnu\_parallel:: \_\_accumulate\_selector< \_It >, 874
- \_\_gnu\_parallel:: \_\_adjacent\_difference\_selector< \_It >, 875
- \_\_gnu\_parallel:: \_\_adjacent\_find\_selector, 876
  - operator(), 877
- \_\_gnu\_parallel:: \_\_binder1st
  - argument\_type, 879
  - result\_type, 879
- \_\_gnu\_parallel:: \_\_binder2nd
  - argument\_type, 881
  - result\_type, 881
- \_\_gnu\_parallel:: \_\_count\_if\_selector
  - operator(), 882
- \_\_gnu\_parallel:: \_\_count\_selector
  - \_M\_finish\_iterator, 884
  - operator(), 883
- \_\_gnu\_parallel:: \_\_count\_selector< \_It, \_Diff >, 883
- \_\_gnu\_parallel:: \_\_fill\_selector
  - \_M\_finish\_iterator, 885
  - operator(), 885
- \_\_gnu\_parallel:: \_\_fill\_selector< \_It >, 884
- \_\_gnu\_parallel:: \_\_find\_first\_of\_selector
  - operator(), 886
- \_\_gnu\_parallel:: \_\_find\_if\_selector, 887
  - operator(), 888
- \_\_gnu\_parallel:: \_\_for\_each\_selector

- operator(), 889
- \_\_gnu\_parallel::\_\_for\_each\_selector< \_It >, 888
- \_\_gnu\_parallel::\_\_generate\_selector
  - \_M\_finish\_iterator, 891
- operator(), 890
- \_\_gnu\_parallel::\_\_generate\_selector< \_It >, 890
- \_\_gnu\_parallel::\_\_generic\_find\_selector, 891
- \_\_gnu\_parallel::\_\_generic\_for\_each\_selector< \_It >, 892
- \_\_gnu\_parallel::\_\_identity\_selector
  - \_M\_finish\_iterator, 894
- operator(), 894
- \_\_gnu\_parallel::\_\_identity\_selector< \_It >, 893
- \_\_gnu\_parallel::\_\_inner\_product\_selector
  - operator(), 896
- \_\_gnu\_parallel::\_\_mismatch\_selector, 898
- operator(), 898
- \_\_gnu\_parallel::\_\_replace\_if\_selector
  - operator(), 903
- \_\_gnu\_parallel::\_\_replace\_selector
  - \_M\_finish\_iterator, 905
  - \_\_new\_val, 905
  - \_\_replace\_selector, 904
- operator(), 905
- \_\_gnu\_parallel::\_\_replace\_selector< \_It, \_Tp >, 904
- \_\_gnu\_parallel::\_\_transform1\_selector
  - operator(), 906
- \_\_gnu\_parallel::\_\_transform1\_selector< \_It >, 905
- \_\_gnu\_parallel::\_\_transform2\_selector
  - operator(), 907
- \_\_gnu\_parallel::\_\_transform2\_selector< \_It >, 907
- \_\_gnu\_parallel::\_\_unary\_negate
  - argument\_type, 909
  - result\_type, 909
- \_\_gnu\_parallel::\_\_unary\_negate< \_Predicate, argument-  
\_type >, 908
- \_\_gnu\_parallel::balanced\_quicksort\_tag, 967
  - \_\_get\_num\_threads, 967
  - set\_num\_threads, 968
- \_\_gnu\_parallel::balanced\_tag, 968
  - \_\_get\_num\_threads, 968
  - set\_num\_threads, 969
- \_\_gnu\_parallel::constant\_size\_blocks\_tag, 969
- \_\_gnu\_parallel::default\_parallel\_tag, 970
  - \_\_get\_num\_threads, 970
  - set\_num\_threads, 970
- \_\_gnu\_parallel::equal\_split\_tag, 971
- \_\_gnu\_parallel::exact\_tag, 972
  - \_\_get\_num\_threads, 972
  - set\_num\_threads, 972
- \_\_gnu\_parallel::find\_tag, 973
- \_\_gnu\_parallel::growing\_blocks\_tag, 974
- \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 974
  - set\_num\_threads, 975
- \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 976
  - set\_num\_threads, 976
- \_\_gnu\_parallel::multiway\_mergesort\_tag, 977
  - \_\_get\_num\_threads, 977
  - set\_num\_threads, 977
- \_\_gnu\_parallel::omp\_loop\_static\_tag, 978
  - set\_num\_threads, 979
- \_\_gnu\_parallel::omp\_loop\_tag, 979
  - \_\_get\_num\_threads, 980
  - set\_num\_threads, 980
- \_\_gnu\_parallel::parallel\_tag, 981
  - \_\_get\_num\_threads, 982
  - parallel\_tag, 982
  - set\_num\_threads, 982
- \_\_gnu\_parallel::quicksort\_tag, 983
  - \_\_get\_num\_threads, 983
  - set\_num\_threads, 983
- \_\_gnu\_parallel::sampling\_tag, 984
  - \_\_get\_num\_threads, 984
  - set\_num\_threads, 984
- \_\_gnu\_parallel::sequential\_tag, 985
- \_\_gnu\_parallel::unbalanced\_tag, 985
  - \_\_get\_num\_threads, 986
  - set\_num\_threads, 986
- gnu\_pbds, 368
- gnu\_pbds::associative\_tag, 986
- gnu\_pbds::basic\_branch< Key, Mapped, Tag, Node\_  
Update, Policy\_Tl, \_Alloc >, 987
- gnu\_pbds::basic\_branch\_tag, 988
- gnu\_pbds::basic\_hash\_tag, 990
- gnu\_pbds::basic\_invalidation\_guarantee, 991
- gnu\_pbds::binary\_heap\_tag, 992
- gnu\_pbds::binomial\_heap\_tag, 993
- gnu\_pbds::cc\_hash\_table
  - cc\_hash\_table, 999d
- gnu\_pbds::cc\_hash\_tag, 1002
- gnu\_pbds::container\_error, 1003
  - what, 1003
- gnu\_pbds::container\_tag, 1004
- gnu\_pbds::container\_traits< Cntnr >, 1005
- gnu\_pbds::container\_traits\_base< binary\_heap\_tag >, 1006
- gnu\_pbds::container\_traits\_base< binomial\_heap\_tag  
>, 1006
- gnu\_pbds::container\_traits\_base< cc\_hash\_tag >, 1007
- gnu\_pbds::container\_traits\_base< gp\_hash\_tag >, 1007
- gnu\_pbds::container\_traits\_base< list\_update\_tag >, 1007
- gnu\_pbds::container\_traits\_base< ov\_tree\_tag >, 1008
- gnu\_pbds::container\_traits\_base< pairing\_heap\_tag  
>, 1008

[\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#),  
[1009](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#), [1009](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_](#)  
[tag >](#), [1009](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#),  
[1010](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#),  
[1010](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it](#)  
[operator\\*](#), [1021](#)  
[operator==](#), [1021](#)  
[reference](#), [1020](#)  
[value\\_type](#), [1020](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#)  
[node\\_const\\_iterator](#), [1024](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn,](#)  
[\\_Alloc >](#), [1026](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_](#)  
[const\\_pointer](#), [1029](#)  
[const\\_reference](#), [1029](#)  
[difference\\_type](#), [1029](#)  
[iterator\\_category](#), [1030](#)  
[operator\\*](#), [1031](#)  
[operator->](#), [1031](#)  
[operator==](#), [1031](#)  
[pointer](#), [1030](#)  
[reference](#), [1030](#)  
[value\\_type](#), [1030](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_](#)  
[operator\\*](#), [1034](#)  
[operator->](#), [1034](#)  
[operator==](#), [1034](#)  
[pointer](#), [1033](#)  
[reference](#), [1033](#)  
[\\_\\_gnu\\_pbds::detail::binomial\\_heap< Value\\_Type, Cmp\\_](#)  
[Fn, \\_Alloc >](#), [1035](#)  
[\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map](#)  
[empty](#), [1044](#)  
[get\\_comb\\_hash\\_fn](#), [1044](#)  
[get\\_eq\\_fn](#), [1044](#), [1045](#)  
[get\\_hash\\_fn](#), [1045](#)  
[get\\_resize\\_policy](#), [1045](#)  
[\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#), [1046](#)  
[\\_\\_gnu\\_pbds::detail::default\\_comb\\_hash\\_fn](#), [1057](#)  
[type](#), [1057](#)  
[\\_\\_gnu\\_pbds::detail::default\\_eq\\_fn](#)  
[type](#), [1058](#)  
[\\_\\_gnu\\_pbds::detail::default\\_eq\\_fn< Key >](#), [1058](#)  
[\\_\\_gnu\\_pbds::detail::default\\_hash\\_fn](#)  
[type](#), [1058](#)  
[\\_\\_gnu\\_pbds::detail::default\\_hash\\_fn< Key >](#), [1058](#)  
[\\_\\_gnu\\_pbds::detail::default\\_probe\\_fn](#)  
[type](#), [1059](#)  
[\\_\\_gnu\\_pbds::detail::default\\_probe\\_fn< Comb\\_Probe\\_Fn](#)  
[>](#), [1059](#)  
[\\_\\_gnu\\_pbds::detail::default\\_resize\\_policy](#)  
[type](#), [1060](#)  
[\\_\\_gnu\\_pbds::detail::default\\_resize\\_policy< Comb\\_Hash\\_](#)  
[Fn >](#), [1059](#)  
[\\_\\_gnu\\_pbds::detail::default\\_update\\_policy](#), [1060](#)  
[type](#), [1061](#)  
[\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator< Key, Data,](#)  
[\\_Alloc >](#), [1061](#)  
[\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, false](#)  
[>](#), [1063](#)  
[\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, true](#)  
[>](#), [1063](#)  
[\\_\\_gnu\\_pbds::detail::eq\\_by\\_less< Key, Cmp\\_Fn >](#), [1063](#)  
[\\_\\_gnu\\_pbds::detail::gp\\_ht\\_map](#)  
[empty](#), [1067](#)  
[get\\_comb\\_probe\\_fn](#), [1067](#)  
[get\\_eq\\_fn](#), [1067](#)  
[get\\_hash\\_fn](#), [1067](#), [1068](#)  
[get\\_probe\\_fn](#), [1068](#)  
[get\\_resize\\_policy](#), [1068](#)  
[\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc,](#)  
[false >](#), [1069](#)  
[\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc,](#)  
[true >](#), [1069](#)  
[\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >](#),  
[1080](#)  
[\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type](#)  
[>](#), [1081](#)  
[\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_](#)  
[Alloc, Update\\_Policy >](#), [1082](#)  
[\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_](#)  
[Type >](#), [1084](#)  
[\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_](#)  
[Type >](#), [1085](#)  
[\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, Mapped >](#),  
[1085](#)  
[\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, null\\_type >](#),  
[1086](#)  
[\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)  
[node\\_begin](#), [1088](#)  
[node\\_end](#), [1089](#)  
[\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it](#)  
[get\\_l\\_child](#), [1094](#)  
[get\\_r\\_child](#), [1094](#)  
[operator\\*](#), [1094](#)  
[\\_\\_gnu\\_pbds::detail::pairing\\_heap< Value\\_Type, Cmp\\_Fn,](#)  
[\\_Alloc >](#), [1094](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#), [1097](#)  
[node\\_type](#), [1098](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Meta-](#)  
[data, \\_Alloc >](#), [1108](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer](#)

[get\\_child](#), [1113](#)  
[operator\\*](#), [1113](#)  
[operator==](#), [1114](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter](#)  
[get\\_child](#), [1116](#)  
[operator\\*](#), [1117](#)  
[operator==](#), [1117](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)  
[node\\_begin](#), [1120](#)  
[node\\_end](#), [1120](#)  
[node\\_type](#), [1120](#)  
[\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base< \\_Alloc >](#), [1121](#)  
[\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map](#)  
[node\\_begin](#), [1129](#)  
[node\\_end](#), [1130](#)  
[\\_\\_gnu\\_pbds::detail::rc< \\_Node, \\_Alloc >](#), [1131](#)  
[\\_\\_gnu\\_pbds::detail::resize\\_policy< \\_Tp >](#), [1135](#)  
[\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map](#)  
[node\\_begin](#), [1139](#)  
[node\\_end](#), [1139](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th >](#), [1141](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, null\\_type >](#), [1142](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_hash< \\_Th >](#), [1143](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_value< \\_Tv >](#), [1144](#)  
[\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, ATraits >](#), [1144](#)  
[\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#), [1145](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_ Update, false >](#), [1147](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_ Update, true >](#), [1148](#)  
[\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_ Update, false >](#), [1161](#)  
[\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_ Update, true >](#), [1162](#)  
[\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, false >](#), [1168](#)  
[\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, true >](#), [1168](#)  
[\\_\\_gnu\\_pbds::detail::type\\_base< Key, null\\_type, \\_Alloc, false >](#), [1169](#)  
[\\_\\_gnu\\_pbds::detail::type\\_base< Key, null\\_type, \\_Alloc, true >](#), [1170](#)  
[\\_\\_gnu\\_pbds::detail::type\\_dispatch< Key, Mapped, \\_Alloc, Store\\_Hash >](#), [1170](#)  
[\\_\\_gnu\\_pbds::detail::types\\_traits< Key, Mapped, \\_Alloc, Store\\_Hash >](#), [1171](#)  
[\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing](#)  
[operator\(\)](#), [1172](#)  
[\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#), [1172](#)  
[\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing](#)  
[operator\(\)](#), [1174](#)  
[\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#), [1173](#)  
[\\_\\_gnu\\_pbds::gp\\_hash\\_table](#)  
[gp\\_hash\\_table](#), [1176d](#)  
[\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#), [1179](#)  
[\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy](#)  
[hash\\_exponential\\_size\\_policy](#), [1180](#)  
[\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#), [1179](#)  
[\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger](#)  
[get\\_loads](#), [1182](#)  
[notify\\_cleared](#), [1182](#)  
[notify\\_inserted](#), [1182](#)  
[notify\\_resized](#), [1182](#)  
[set\\_loads](#), [1182](#)  
[\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#), [1182](#)  
[hash\\_prime\\_size\\_policy](#), [1183](#)  
[size\\_type](#), [1183](#)  
[\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy](#)  
[get\\_actual\\_size](#), [1185](#)  
[get\\_new\\_size](#), [1185](#)  
[get\\_size\\_policy](#), [1185](#)  
[get\\_trigger\\_policy](#), [1185](#)  
[hash\\_standard\\_resize\\_policy](#), [1184](#)  
[resize](#), [1186](#)  
[\\_\\_gnu\\_pbds::insert\\_error](#), [1186](#)  
[what](#), [1187](#)  
[\\_\\_gnu\\_pbds::join\\_error](#), [1187](#)  
[what](#), [1188](#)  
[\\_\\_gnu\\_pbds::linear\\_probe\\_fn](#)  
[operator\(\)](#), [1188](#)  
[\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#), [1188](#)  
[\\_\\_gnu\\_pbds::list\\_update](#)  
[list\\_update](#), [1189](#)  
[\\_\\_gnu\\_pbds::list\\_update< Key, Mapped, Eq\\_Fn, Update\\_Policy, \\_Alloc >](#), [1189](#)  
[\\_\\_gnu\\_pbds::list\\_update\\_tag](#), [1190](#)  
[\\_\\_gnu\\_pbds::lu\\_counter\\_policy](#)  
[metadata\\_reference](#), [1192](#)  
[metadata\\_type](#), [1192](#)  
[operator\(\)](#), [1192](#)  
[\\_\\_gnu\\_pbds::lu\\_counter\\_policy< Max\\_Count, \\_Alloc >](#), [1191](#)  
[\\_\\_gnu\\_pbds::lu\\_move\\_to\\_front\\_policy](#)  
[metadata\\_reference](#), [1193](#)  
[metadata\\_type](#), [1193](#)  
[operator\(\)](#), [1193](#)  
[\\_\\_gnu\\_pbds::lu\\_move\\_to\\_front\\_policy< \\_Alloc >](#), [1193](#)  
[\\_\\_gnu\\_pbds::null\\_type](#), [1195](#)  
[\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#), [1196](#)  
[\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#), [1197](#)  
[\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#), [1198](#)  
[\\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#), [1199](#)

- \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc  
>, 1199
- \_\_gnu\_pbds::priority\_queue\_tag, 1201
- \_\_gnu\_pbds::quadratic\_probe\_fn  
operator(), 1202
- \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type >, 1201
- \_\_gnu\_pbds::range\_invalidation\_guarantee, 1202
- \_\_gnu\_pbds::rb\_tree\_tag, 1203
- \_\_gnu\_pbds::rc\_binomial\_heap\_tag, 1204
- \_\_gnu\_pbds::resize\_error, 1205
  - what, 1205
- \_\_gnu\_pbds::sample\_probe\_fn, 1206
  - operator(), 1206
  - sample\_probe\_fn, 1206
  - swap, 1206
- \_\_gnu\_pbds::sample\_range\_hashing, 1206
  - notify\_resized, 1207
  - operator(), 1207
  - sample\_range\_hashing, 1207
  - size\_type, 1207
  - swap, 1207
- \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 1208
  - notify\_resized, 1208
  - operator(), 1208
  - sample\_ranged\_hash\_fn, 1208
  - swap, 1208
- \_\_gnu\_pbds::sample\_ranged\_probe\_fn, 1209
- \_\_gnu\_pbds::sample\_resize\_policy, 1209
  - get\_new\_size, 1210
  - is\_resize\_needed, 1210
  - notify\_cleared, 1210
  - notify\_erase\_search\_collision, 1210
  - notify\_erase\_search\_end, 1210
  - notify\_erase\_search\_start, 1210
  - notify\_erased, 1210
  - notify\_find\_search\_collision, 1211
  - notify\_find\_search\_end, 1211
  - notify\_find\_search\_start, 1211
  - notify\_insert\_search\_collision, 1211
  - notify\_insert\_search\_end, 1211
  - notify\_insert\_search\_start, 1211
  - notify\_inserted, 1211
  - notify\_resized, 1211
  - sample\_range\_hashing, 1211
  - sample\_resize\_policy, 1210
  - size\_type, 1210
  - swap, 1211
- \_\_gnu\_pbds::sample\_resize\_trigger, 1211
  - is\_grow\_needed, 1212
  - is\_resize\_needed, 1212
  - notify\_cleared, 1213
  - notify\_erase\_search\_collision, 1213
  - notify\_erase\_search\_end, 1213
  - notify\_erase\_search\_start, 1213
- notify\_erased, 1213
- notify\_externally\_resized, 1213
- notify\_find\_search\_collision, 1213
- notify\_find\_search\_end, 1213
- notify\_find\_search\_start, 1213
- notify\_insert\_search\_collision, 1213
- notify\_insert\_search\_end, 1213
- notify\_insert\_search\_start, 1213
- notify\_inserted, 1213
- notify\_resized, 1213
- sample\_range\_hashing, 1214
- sample\_resize\_trigger, 1212
- size\_type, 1212
- swap, 1214
- \_\_gnu\_pbds::sample\_size\_policy, 1214
  - get\_nearest\_larger\_size, 1215
  - get\_nearest\_smaller\_size, 1215
  - sample\_range\_hashing, 1215
  - sample\_size\_policy, 1214
  - size\_type, 1214
  - swap, 1215
- \_\_gnu\_pbds::sample\_trie\_access\_traits, 1215
  - begin, 1216
  - e\_pos, 1216
  - e\_type, 1216
  - end, 1216
- \_\_gnu\_pbds::sample\_trie\_node\_update
  - operator(), 1217
  - sample\_trie\_node\_update, 1217
- \_\_gnu\_pbds::sample\_update\_policy, 1217
  - metadata\_type, 1218
  - operator(), 1218
  - sample\_update\_policy, 1218
  - swap, 1218
- \_\_gnu\_pbds::sequence\_tag, 1219
- \_\_gnu\_pbds::splay\_tree\_tag, 1220
- \_\_gnu\_pbds::string\_tag, 1221
- \_\_gnu\_pbds::thin\_heap\_tag, 1222
- \_\_gnu\_pbds::tree
  - cmp\_fn, 1223
  - tree, 1223, 1224
- \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_  
Update, \_Alloc >, 1222
- \_\_gnu\_pbds::tree\_order\_statistics\_node\_update
  - find\_by\_order, 1226
  - operator(), 1226
  - order\_of\_key, 1226
- \_\_gnu\_pbds::tree\_tag, 1227
- \_\_gnu\_pbds::trie
  - access\_traits, 1228
  - trie, 1229
- \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_  
Update, \_Alloc >, 1227
- \_\_gnu\_pbds::trie\_order\_statistics\_node\_update

- find\_by\_order, 1232
- operator(), 1232
- order\_of\_key, 1232
- order\_of\_prefix, 1232
- \_\_gnu\_pbds::trie\_prefix\_search\_node\_update
  - a\_const\_iterator, 1235
  - access\_traits, 1235
  - allocator\_type, 1235
  - operator(), 1235
  - prefix\_range, 1235, 1236
  - size\_type, 1235
- \_\_gnu\_pbds::trie\_string\_access\_traits
  - begin, 1237
  - const\_iterator, 1237
  - e\_pos, 1237
  - e\_type, 1237
  - end, 1238
- \_\_gnu\_pbds::trie\_tag, 1238
- \_\_gnu\_pbds::trivial\_iterator\_tag, 1239
- \_\_gnu\_profile, 371
  - \_\_env\_t, 375
  - \_\_profcxx\_init, 376
  - \_\_report, 376
- \_\_gnu\_profile::\_\_container\_size\_info, 1239
- \_\_gnu\_profile::\_\_container\_size\_stack\_info, 1240
- \_\_gnu\_profile::\_\_hashfunc\_info, 1241
- \_\_gnu\_profile::\_\_hashfunc\_stack\_info, 1242
- \_\_gnu\_profile::\_\_list2vector\_info, 1243
- \_\_gnu\_profile::\_\_map2umap\_info, 1245
- \_\_gnu\_profile::\_\_map2umap\_stack\_info, 1246
- \_\_gnu\_profile::\_\_object\_info\_base, 1247
- \_\_gnu\_profile::\_\_reentrance\_guard, 1248
- \_\_gnu\_profile::\_\_stack\_hash, 1248
- \_\_gnu\_profile::\_\_trace\_container\_size, 1250
- \_\_gnu\_profile::\_\_trace\_hash\_func, 1251
- \_\_gnu\_profile::\_\_trace\_hashtable\_size, 1252
- \_\_gnu\_profile::\_\_trace\_map2umap, 1253
- \_\_gnu\_profile::\_\_trace\_vector\_size, 1254
- \_\_gnu\_profile::\_\_trace\_vector\_to\_list, 1255
- \_\_gnu\_profile::\_\_vector2list\_info, 1256
- \_\_gnu\_profile::\_\_vector2list\_stack\_info, 1257
- \_\_gnu\_profile::\_\_warning\_data, 1258
- \_\_gnu\_sequential, 376
- \_\_heap\_select
  - std, 499, 500
- \_\_init\_winner
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 930
- \_\_inner\_product\_selector
  - \_\_gnu\_parallel::\_\_inner\_product\_selector, 895
- \_\_inplace\_stable\_partition
  - std, 500
- \_\_inplace\_stable\_sort
  - std, 500
- \_\_insert\_start
  - \_\_gnu\_parallel::\_\_LoserTree, 927
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 930
  - \_\_gnu\_parallel::\_\_LoserTreeBase, 933
- \_\_insertion\_sort
  - std, 500
- \_\_introsort\_loop
  - std, 500, 501
- \_\_invoke
  - std, 501
- \_\_ioint
  - std, 565
- \_\_is\_sorted
  - \_\_gnu\_parallel, 342
- \_\_iterator\_category
  - Iterators, 233
- \_\_lg
  - std, 501
- \_\_match\_flag
  - std::regex\_constants, 612
- \_\_median
  - SGI, 10
- \_\_median\_of\_three\_iterators
  - \_\_gnu\_parallel, 342
- \_\_merge\_adaptive
  - std, 501
- \_\_merge\_advance
  - \_\_gnu\_parallel, 342
- \_\_merge\_advance\_movc
  - \_\_gnu\_parallel, 343
- \_\_merge\_advance\_usual
  - \_\_gnu\_parallel, 343
- \_\_merge\_without\_buffer
  - std, 501, 502
- \_\_move\_median\_first
  - std, 502
- \_\_move\_merge
  - std, 502
- \_\_move\_merge\_adaptive
  - std, 502, 503
- \_\_move\_merge\_adaptive\_backward
  - std, 503
- \_\_new\_val
  - \_\_gnu\_parallel::\_\_replace\_if\_selector, 903
  - \_\_gnu\_parallel::\_\_replace\_selector, 905
- \_\_num\_bitmaps
  - \_\_gnu\_cxx::\_\_detail, 313
- \_\_num\_blocks
  - \_\_gnu\_cxx::\_\_detail, 313
- \_\_num\_get\_type
  - std::basic\_ios, 1661
  - std::basic\_ofstream, 1811
  - std::basic\_ostream, 1844

- std::basic\_ostringstream, 1877
- \_\_num\_put\_type
  - std::basic\_fstream, 1576
  - std::basic\_ifstream, 1623
  - std::basic\_ios, 1661
  - std::basic\_iostream, 1687
  - std::basic\_istream, 1732
  - std::basic\_istreamstream, 1773
  - std::basic\_stringstream, 1998
- \_\_parallel\_merge\_advance
  - \_\_gnu\_parallel, 344
- \_\_parallel\_nth\_element
  - \_\_gnu\_parallel, 345
- \_\_parallel\_partial\_sort
  - \_\_gnu\_parallel, 345
- \_\_parallel\_partial\_sum
  - \_\_gnu\_parallel, 345
- \_\_parallel\_partial\_sum\_basecase
  - \_\_gnu\_parallel, 346
- \_\_parallel\_partial\_sum\_linear
  - \_\_gnu\_parallel, 346
- \_\_parallel\_partition
  - \_\_gnu\_parallel, 347
- \_\_parallel\_random\_shuffle
  - \_\_gnu\_parallel, 347
- \_\_parallel\_random\_shuffle\_drs
  - \_\_gnu\_parallel, 347
- \_\_parallel\_random\_shuffle\_drs\_pu
  - \_\_gnu\_parallel, 348
- \_\_parallel\_sort
  - \_\_gnu\_parallel, 348d
- \_\_parallel\_sort\_qs
  - \_\_gnu\_parallel, 352
- \_\_parallel\_sort\_qs\_conquer
  - \_\_gnu\_parallel, 353
- \_\_parallel\_sort\_qs\_divide
  - \_\_gnu\_parallel, 353
- \_\_parallel\_sort\_qsb
  - \_\_gnu\_parallel, 353
- \_\_parallel\_unique\_copy
  - \_\_gnu\_parallel, 354
- \_\_partition
  - std, 503
- \_\_pool< \_Thread >, 1259
- \_\_profcxx\_init
  - \_\_gnu\_profile, 376
- \_\_qsb\_conquer
  - \_\_gnu\_parallel, 355
- \_\_qsb\_divide
  - \_\_gnu\_parallel, 355
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, 355
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, 356
- \_\_rd\_log2
  - \_\_gnu\_parallel, 356
- \_\_rebind\_m
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 1112
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 1116
- \_\_reflection\_typelist< \_Elements >, 1259
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_replace\_if\_selector, 902
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_replace\_selector, 904
- \_\_report
  - \_\_gnu\_profile, 376
- \_\_reverse
  - std, 503
- \_\_rotate
  - std, 504
- \_\_rotate\_adaptive
  - std, 504
- \_\_round\_up\_to\_pow2
  - \_\_gnu\_parallel, 356
- \_\_search\_n
  - std, 504, 505
- \_\_search\_template
  - \_\_gnu\_parallel, 356
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 357
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 357
- \_\_shrink
  - \_\_gnu\_parallel, 358
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 358
- \_\_stable\_partition\_adaptive
  - std, 505
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 303
- \_\_streambuf\_type
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 787
  - std::basic\_streambuf, 1915
- \_\_syntax\_option
  - std::regex\_constants, 612
- \_\_umap\_traits
  - std, 496
- \_\_ummap\_traits
  - std, 496
- \_\_umset\_traits
  - std, 496
- \_\_unguarded\_insertion\_sort
  - std, 505
- \_\_unguarded\_linear\_insert
  - std, 505, 506
- \_\_unguarded\_partition
  - std, 506

- \_\_unguarded\_partition\_pivot
  - std, 506
- \_\_unique\_copy
  - std, 506, 507
- \_\_uset\_traits
  - std, 496
- \_\_valid\_range
  - \_\_gnu\_debug, 322
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 322
- \_\_valid\_range\_aux2
  - \_\_gnu\_debug, 322
- \_\_verbose\_terminate\_handler
  - Exceptions, 25
- \_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string, 651d
- \_\_yield
  - \_\_gnu\_parallel, 358
- a
  - std::extreme\_value\_distribution, 2217
  - std::weibull\_distribution, 2886
- a\_const\_iterator
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 1235
- abi, 376
- abs
  - Complex Numbers, 31
- access\_traits
  - \_\_gnu\_pbds::trie, 1228
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 1235
- accumulate
  - std, 508
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 961
- acos
  - std, 508
- acosh
  - Complex Numbers, 32
  - std, 508
- Adaptors for pointers to functions, 221
  - ptr\_fun, 221
- Adaptors for pointers to members, 223
- addressof
  - Utilities, 67
- adjacent\_difference
  - std, 509
- adjacent\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, 961
- adjacent\_find
  - Non-Mutating, 122
- adjustfield
  - std::basic\_fstream, 1613
  - std::basic\_ifstream, 1653
  - std::basic\_ios, 1676
  - std::basic\_iostream, 1723
  - std::basic\_istream, 1761
  - std::basic\_istream, 1802
  - std::basic\_ofstream, 1835
- advance
  - std, 509
- algo.h, 2892
- algorithbase.h, 2902
- algorithm, 2903, 2904
- algorithmfwd.h, 2904, 2910
- Algorithms, 103
- all
  - std, 510
  - std::locale, 2415
  - std::tr2::dynamic\_bitset, 2753
- all\_of
  - Non-Mutating, 123
- alloc\_traits.h, 2919, 2920
- allocate
  - \_\_gnu\_cxx::\_\_alloc\_traits, 630
  - std::allocator\_traits, 1498
- allocate\_shared
  - Pointer Abstractions, 50
  - std::shared\_ptr, 2698
- allocator.h, 2920
- allocator\_type
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 1235
  - std::allocator\_traits, 1496
  - std::set, 2679
  - std::unordered\_map, 2786
  - std::unordered\_multimap, 2806
  - std::unordered\_multiset, 2825
  - std::unordered\_set, 2842
- Allocators, 161
  - \_\_allocator\_base, 162
- alpha
  - std::gamma\_distribution, 2253
- any
  - std, 510
  - std::tr2::dynamic\_bitset, 2753
- any\_of
  - Non-Mutating, 123
- app
  - std::basic\_fstream, 1613
  - std::basic\_ifstream, 1653
  - std::basic\_ios, 1676
  - std::basic\_iostream, 1723
  - std::basic\_istream, 1761
  - std::basic\_istream, 1802
  - std::basic\_ofstream, 1835

- std::basic\_ostream, 1866
- std::basic\_ostringstream, 1899
- std::basic\_stringstream, 2035
- std::ios\_base, 2314
- append
  - \_\_gnu\_cxx::\_\_versa\_string, 654d
  - \_\_gnu\_debug::basic\_string, 858, 859
  - std::basic\_string, 1937d
  - std::tr2::dynamic\_bitset, 2753
- apply
  - Numeric Arrays, 85, 86
- apply\_generator
  - \_\_gnu\_cxx::typelist, 314
- arg
  - Complex Numbers, 32
  - std, 510
- argument\_type
  - \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder, 635
  - \_\_gnu\_cxx::binary\_compose, 706
  - \_\_gnu\_cxx::select1st, 759
  - \_\_gnu\_cxx::select2nd, 760
  - \_\_gnu\_cxx::subtractive\_rng, 801
  - \_\_gnu\_cxx::unary\_compose, 812
  - \_\_gnu\_parallel::\_\_binder1st, 879
  - \_\_gnu\_parallel::\_\_binder2nd, 881
  - \_\_gnu\_parallel::\_\_unary\_negate, 909
  - std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1 >, 1468
  - std::binder1st, 2048
  - std::binder2nd, 2049
  - std::const\_mem\_fun\_ref\_t, 2106
  - std::const\_mem\_fun\_t, 2108
  - std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2273
  - std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2274
  - std::logical\_not, 2424
  - std::mem\_fun\_ref\_t, 2462
  - std::mem\_fun\_t, 2463
  - std::negate, 2535
  - std::pointer\_to\_unary\_function, 2631
  - std::unary\_function, 2771
  - std::unary\_negate, 2773
- Arithmetic Classes, 216
- array, 2921
- array\_allocator.h, 2922
- asin
  - std, 510
- asinh
  - Complex Numbers, 32
  - std, 510
- assign
  - \_\_gnu\_cxx::\_\_versa\_string, 656d
  - \_\_gnu\_debug::basic\_string, 859, 860
  - std::\_\_detail::\_Nfa, 1395, 1396
  - std::basic\_regex, 1908d
  - std::basic\_string, 1940d
  - std::deque, 2183
  - std::forward\_list, 2229
  - std::list, 2396, 2397
  - std::vector, 2870
- assoc\_container.hpp, 2922
- assoc\_laguerre
  - Mathematical Special Functions, 243
- assoc\_legendre
  - Mathematical Special Functions, 243
- Associative, 19
- at
  - \_\_gnu\_cxx::\_\_versa\_string, 659, 660
  - \_\_gnu\_debug::basic\_string, 860
  - std::\_\_detail::\_Nfa, 1396
  - std::basic\_string, 1943
  - std::deque, 2184
  - std::map, 2435
  - std::unordered\_map, 2790
  - std::vector, 2871
- atan
  - std, 511
- atanh
  - Complex Numbers, 32
  - std, 511
- ate
  - std::basic\_fstream, 1613
  - std::basic\_ifstream, 1653
  - std::basic\_ios, 1676
  - std::basic\_iostream, 1723
  - std::basic\_istream, 1761
  - std::basic\_istreamstream, 1802
  - std::basic\_ofstream, 1835
  - std::basic\_ostream, 1866
  - std::basic\_ostringstream, 1899
  - std::basic\_stringstream, 2035
  - std::ios\_base, 2314
- atomic, 2923
- atomic\_base.h, 2927
- atomic\_char
  - Atomics, 168
- atomic\_char16\_t
  - Atomics, 168
- atomic\_char32\_t
  - Atomics, 168
- atomic\_int
  - Atomics, 168
- atomic\_int\_fast16\_t
  - Atomics, 168
- atomic\_int\_fast32\_t
  - Atomics, 169
- atomic\_int\_fast64\_t
  - Atomics, 169

- atomic\_int\_fast8\_t
  - Atomics, 169
- atomic\_int\_least16\_t
  - Atomics, 169
- atomic\_int\_least32\_t
  - Atomics, 169
- atomic\_int\_least64\_t
  - Atomics, 169
- atomic\_int\_least8\_t
  - Atomics, 169
- atomic\_intmax\_t
  - Atomics, 169
- atomic\_intptr\_t
  - Atomics, 169
- atomic\_llong
  - Atomics, 169
- atomic\_lockfree\_defines.h, 2929
- atomic\_long
  - Atomics, 170
- atomic\_ptrdiff\_t
  - Atomics, 170
- atomic\_schar
  - Atomics, 170
- atomic\_short
  - Atomics, 170
- atomic\_size\_t
  - Atomics, 170
- atomic\_uchar
  - Atomics, 170
- atomic\_uint
  - Atomics, 170
- atomic\_uint\_fast16\_t
  - Atomics, 170
- atomic\_uint\_fast32\_t
  - Atomics, 170
- atomic\_uint\_fast64\_t
  - Atomics, 170
- atomic\_uint\_fast8\_t
  - Atomics, 171
- atomic\_uint\_least16\_t
  - Atomics, 171
- atomic\_uint\_least32\_t
  - Atomics, 171
- atomic\_uint\_least64\_t
  - Atomics, 171
- atomic\_uint\_least8\_t
  - Atomics, 171
- atomic\_uintmax\_t
  - Atomics, 171
- atomic\_uintptr\_t
  - Atomics, 171
- atomic\_ullong
  - Atomics, 171
- atomic\_ulong
  - Atomics, 171
- Atomics, 171
- atomic\_ushort
  - Atomics, 171
- atomic\_wchar\_t
  - Atomics, 172
- atomic\_word.h, 2929
- atomicity.h, 2929
- Atomics, 163
  - atomic\_char, 168
  - atomic\_char16\_t, 168
  - atomic\_char32\_t, 168
  - atomic\_int, 168
  - atomic\_int\_fast16\_t, 168
  - atomic\_int\_fast32\_t, 169
  - atomic\_int\_fast64\_t, 169
  - atomic\_int\_fast8\_t, 169
  - atomic\_int\_least16\_t, 169
  - atomic\_int\_least32\_t, 169
  - atomic\_int\_least64\_t, 169
  - atomic\_int\_least8\_t, 169
  - atomic\_intmax\_t, 169
  - atomic\_intptr\_t, 169
  - atomic\_llong, 169
  - atomic\_long, 170
  - atomic\_ptrdiff\_t, 170
  - atomic\_schar, 170
  - atomic\_short, 170
  - atomic\_size\_t, 170
  - atomic\_uchar, 170
  - atomic\_uint, 170
  - atomic\_uint\_fast16\_t, 170
  - atomic\_uint\_fast32\_t, 170
  - atomic\_uint\_fast64\_t, 170
  - atomic\_uint\_fast8\_t, 171
  - atomic\_uint\_least16\_t, 171
  - atomic\_uint\_least32\_t, 171
  - atomic\_uint\_least64\_t, 171
  - atomic\_uint\_least8\_t, 171
  - atomic\_uintmax\_t, 171
  - atomic\_uintptr\_t, 171
  - atomic\_ullong, 171
  - atomic\_ulong, 171
  - atomic\_ushort, 171
  - atomic\_wchar\_t, 172
  - kill\_dependency, 172
  - memory\_order, 172
- auto\_ptr
  - std::auto\_ptr, 1536, 1537
- auto\_ptr.h, 2930
- awk
  - std::regex\_constants, 613
- b
  - std::extreme\_value\_distribution, 2217

- std::weibull\_distribution, 2886
- back
  - \_\_gnu\_cxx::\_\_versa\_string, 660
  - \_\_gnu\_debug::basic\_string, 861
  - std::\_\_detail::\_Nfa, 1397
  - std::basic\_string, 1943
  - std::deque, 2184, 2185
  - std::list, 2397
  - std::queue, 2642
  - std::vector, 2871, 2872
- back\_insert\_iterator
  - std::back\_insert\_iterator, 1541
- back\_inserter
  - Iterators, 233
- backward\_warning.h, 2931
- bad
  - std::basic\_fstream, 1580
  - std::basic\_ifstream, 1626
  - std::basic\_ios, 1665
  - std::basic\_iostream, 1690
  - std::basic\_istream, 1735
  - std::basic\_istreamstream, 1776
  - std::basic\_ofstream, 1815
  - std::basic\_ostream, 1847
  - std::basic\_ostreamstream, 1880
  - std::basic\_stringstream, 2002
- badbit
  - std::basic\_fstream, 1613
  - std::basic\_ifstream, 1653
  - std::basic\_ios, 1676
  - std::basic\_iostream, 1723
  - std::basic\_istream, 1761
  - std::basic\_istreamstream, 1802
  - std::basic\_ofstream, 1835
  - std::basic\_ostream, 1866
  - std::basic\_ostreamstream, 1899
  - std::basic\_stringstream, 2035
  - std::ios\_base, 2314
- balanced\_quicksort.h, 2931
- base
  - \_\_gnu\_debug::\_Safe\_iterator, 820
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 832
  - std::discard\_block\_engine, 2196
  - std::independent\_bits\_engine, 2293
  - std::reverse\_iterator, 2670
  - std::shuffle\_order\_engine, 2701
- Base and Implementation Classes, 175, 211
  - \_AnyMatcher, 213
  - \_AutomatonPtr, 212
  - \_Matcher, 212
  - \_Opcode, 213
  - \_S\_invalid\_state\_id, 213
  - \_StateIdT, 213
  - \_StateSet, 213
  - \_StateStack, 213
  - \_Tagger, 213
- Base and Policy Classes, 250, 252, 257
- base.h, 2931, 2932
- basefield
  - std::basic\_fstream, 1613
  - std::basic\_ifstream, 1653
  - std::basic\_ios, 1677
  - std::basic\_iostream, 1723
  - std::basic\_istream, 1762
  - std::basic\_istreamstream, 1802
  - std::basic\_ofstream, 1835
  - std::basic\_ostream, 1866
  - std::basic\_ostreamstream, 1900
  - std::basic\_stringstream, 2035
  - std::ios\_base, 2314
- basic
  - std::regex\_constants, 614
- basic\_file.h, 2933
- basic\_filebuf
  - std::basic\_filebuf, 1552
- basic\_fstream
  - std::basic\_fstream, 1578, 1579
- basic\_ifstream
  - std::basic\_ifstream, 1625
- basic\_ios
  - std::basic\_ios, 1664
- basic\_ios.h, 2933
- basic\_ios.tcc, 2934
- basic\_iostream
  - std::basic\_iostream, 1689
- basic\_istream
  - std::basic\_istream, 1735
- basic\_istreamstream
  - std::basic\_istreamstream, 1775
- basic\_iterator.h, 2934
- basic\_ofstream
  - std::basic\_ofstream, 1814
- basic\_ostream
  - std::basic\_ostream, 1847
- basic\_ostreamstream
  - std::basic\_ostreamstream, 1879
- basic\_regex
  - std::basic\_regex, 1905d
- basic\_streambuf
  - std::basic\_streambuf, 1916
- basic\_string
  - std::basic\_string, 1935d
- basic\_string.h, 2934
- basic\_string.tcc, 2937
- basic\_stringbuf
  - std::basic\_stringbuf, 1977
- basic\_stringstream
  - std::basic\_stringstream, 2001

- before\_begin
  - std::forward\_list, 2230
- beg
  - std::basic\_fstream, 1614
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1677
  - std::basic\_iostream, 1723
  - std::basic\_istream, 1762
  - std::basic\_istreamstream, 1803
  - std::basic\_ofstream, 1835
  - std::basic\_ostream, 1866
  - std::basic\_ostreamstream, 1900
  - std::basic\_stringstream, 2036
  - std::ios\_base, 2314
- begin
  - \_\_gnu\_cxx::\_\_versa\_string, 661
  - \_\_gnu\_cxx::temporary\_buffer, 803
  - \_\_gnu\_parallel::\_PseudoSequence, 952
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1216
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 1237
  - Numeric Arrays, 86
  - std, 511
  - std::\_\_detail::\_Nfa, 1397
  - std::\_Temporary\_buffer, 1480
  - std::basic\_string, 1944
  - std::deque, 2185
  - std::forward\_list, 2230
  - std::list, 2398
  - std::map, 2435
  - std::match\_results, 2453
  - std::multimap, 2508
  - std::multiset, 2524
  - std::set, 2683
  - std::unordered\_map, 2791
  - std::unordered\_multimap, 2810, 2811
  - std::unordered\_multiset, 2828, 2829
  - std::unordered\_set, 2846
  - std::vector, 2872
- Bernoulli Distributions, 278
  - operator<<, 279
  - operator>>, 280
- bernoulli\_distribution
  - std::bernoulli\_distribution, 2041
- beta
  - Mathematical Special Functions, 243
  - std::gamma\_distribution, 2253
- bin\_search\_tree.hpp, 2938
- binary
  - std::basic\_fstream, 1614
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1677
  - std::basic\_iostream, 1724
  - std::basic\_istream, 1762
  - std::basic\_istreamstream, 1803
  - std::basic\_ofstream, 1836
  - std::basic\_ostream, 1867
  - std::basic\_ostreamstream, 1900
  - std::basic\_stringstream, 2036
  - std::ios\_base, 2314
- Binary Search, 156
  - binary\_search, 157
  - equal\_range, 157, 158
  - lower\_bound, 158, 159
  - upper\_bound, 159
- binary\_heap.hpp, 2938
- binary\_heap\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 1030
- binary\_search
  - Binary Search, 157
- bind
  - Binder Classes, 239
- bind1st
  - Binder Classes, 239
- bind2nd
  - Binder Classes, 240
- Binder Classes, 238
  - bind, 239
  - bind1st, 239
  - bind2nd, 240
- binders.h, 2939
- binomial\_heap.hpp, 2939
- binomial\_heap\_base.hpp, 2940
- bitmap\_allocator.h, 2940
- bitset, 2941, 2944
- bool\_set, 2945
  - std::tr2::bool\_set, 2746
- bool\_set.tcc, 2946
- boolalpha
  - std, 512
  - std::basic\_fstream, 1614
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1677
  - std::basic\_iostream, 1724
  - std::basic\_istream, 1762
  - std::basic\_istreamstream, 1803
  - std::basic\_ofstream, 1836
  - std::basic\_ostream, 1867
  - std::basic\_ostreamstream, 1900
  - std::basic\_stringstream, 2036
  - std::ios\_base, 2315
- Boolean Operations Classes, 218
- boost\_concept\_check.h, 2946
- Branch-Based, 251
- branch\_policy.hpp, 2947
- bucket
  - \_\_gnu\_debug::\_Safe\_local\_iterator, 832
- bucket\_count

- [std::unordered\\_map](#), [2792](#)
  - [std::unordered\\_multimap](#), [2811](#)
  - [std::unordered\\_multiset](#), [2829](#)
  - [std::unordered\\_set](#), [2847](#)
- [c](#)
  - [std::queue](#), [2643](#)
- [c++0x\\_warning.h](#), [2948](#)
- [c++allocator.h](#), [2948](#)
- [c++config.h](#), [2948](#)
- [c++io.h](#), [2953](#)
- [c++locale.h](#), [2953](#)
- [c++locale\\_internal.h](#), [2954](#)
- [c\\_str](#)
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#), [661](#)
  - [std::basic\\_string](#), [1944](#)
- [cache\\_line\\_size](#)
  - [\\_\\_gnu\\_parallel::\\_Settings](#), [961](#)
- [call\\_once](#)
  - [Mutexes](#), [53](#)
  - [std::once\\_flag](#), [2606](#)
- [capacity](#)
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#), [661](#)
  - [\\_\\_gnu\\_debug::basic\\_string](#), [861](#)
  - [std::\\_\\_detail::\\_Nfa](#), [1397](#)
  - [std::basic\\_string](#), [1944](#)
  - [std::vector](#), [2872](#)
- [cassert](#), [2954](#)
- [cast.h](#), [2954](#)
- [category](#)
  - [std::locale](#), [2410](#)
- [cbefore\\_begin](#)
  - [std::forward\\_list](#), [2230](#)
- [cbegin](#)
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#), [661](#)
  - [std::\\_\\_detail::\\_Nfa](#), [1397](#)
  - [std::basic\\_string](#), [1944](#)
  - [std::deque](#), [2185](#)
  - [std::forward\\_list](#), [2230](#)
  - [std::list](#), [2398](#)
  - [std::map](#), [2435](#)
  - [std::match\\_results](#), [2453](#)
  - [std::multimap](#), [2509](#)
  - [std::multiset](#), [2524](#)
  - [std::set](#), [2683](#)
  - [std::unordered\\_map](#), [2792](#)
  - [std::unordered\\_multimap](#), [2811](#)
  - [std::unordered\\_multiset](#), [2829](#)
  - [std::unordered\\_set](#), [2847](#)
  - [std::vector](#), [2872](#)
- [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger\\_imp.hpp](#), [2955](#)
- [cc\\_hash\\_table](#)
  - [\\_\\_gnu\\_pbds::cc\\_hash\\_table](#), [999d](#)
- [cc\\_ht\\_map.hpp](#), [2955](#)
- [ccomplex](#), [2955](#), [2956](#)
- [cctype](#), [2956](#)
- [cend](#)
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#), [661](#)
  - [std::\\_\\_detail::\\_Nfa](#), [1397](#)
  - [std::basic\\_string](#), [1944](#)
  - [std::deque](#), [2185](#)
  - [std::forward\\_list](#), [2231](#)
  - [std::list](#), [2398](#)
  - [std::map](#), [2436](#)
  - [std::match\\_results](#), [2453](#)
  - [std::multimap](#), [2509](#)
  - [std::multiset](#), [2525](#)
  - [std::set](#), [2683](#)
  - [std::unordered\\_map](#), [2792](#)
  - [std::unordered\\_multimap](#), [2812](#)
  - [std::unordered\\_multiset](#), [2830](#)
  - [std::unordered\\_set](#), [2847](#)
  - [std::vector](#), [2872](#)
- [cerr](#)
  - [std](#), [565](#)
- [cerrno](#), [2957](#)
- [cfenv](#), [2957](#)
- [cfloat](#), [2957](#), [2958](#)
- [char\\_traits.h](#), [2958](#)
- [char\\_type](#)
  - [std::\\_\\_ctype\\_abstract\\_base](#), [1293](#)
  - [std::basic\\_ios](#), [1662](#)
  - [std::basic\\_streambuf](#), [1915](#)
  - [std::collate](#), [2089](#)
  - [std::collate\\_byname](#), [2094](#)
  - [std::ctype< char >](#), [2122](#)
  - [std::ctype< wchar\\_t >](#), [2133](#)
  - [std::ctype\\_byname< char >](#), [2158](#)
  - [std::istreambuf\\_iterator](#), [2377](#)
  - [std::messages](#), [2469](#)
  - [std::money\\_get](#), [2479](#)
  - [std::money\\_put](#), [2483](#)
  - [std::moneypunct](#), [2488](#)
  - [std::num\\_get](#), [2548](#)
  - [std::num\\_put](#), [2562](#)
  - [std::numpunct](#), [2596](#)
  - [std::ostream\\_iterator](#), [2607](#)
  - [std::ostreambuf\\_iterator](#), [2610](#)
  - [std::time\\_get](#), [2720](#)
  - [std::time\\_put](#), [2736](#)
- [checkers.h](#), [2959](#)
- [chrono](#), [2959](#)
- [cin](#)
  - [std](#), [565](#)
- [cinttypes](#), [2962](#)
- [ciso646](#), [2962](#)
- [classic](#)

- std::locale, 2412
- classic\_table
  - std::ctype< char >, 2122
  - std::ctype\_byname< char >, 2158
- clear
  - \_\_gnu\_cxx::\_\_versa\_string, 662
  - std::\_\_detail::\_Nfa, 1398
  - std::basic\_fstream, 1580
  - std::basic\_ifstream, 1626
  - std::basic\_ios, 1665
  - std::basic\_iostream, 1690
  - std::basic\_istream, 1735
  - std::basic\_istreamstream, 1776
  - std::basic\_ofstream, 1815
  - std::basic\_ostream, 1848
  - std::basic\_ostreamstream, 1881
  - std::basic\_string, 1944
  - std::basic\_stringstream, 2002
  - std::deque, 2185
  - std::forward\_list, 2231
  - std::list, 2398
  - std::map, 2436
  - std::multimap, 2509
  - std::multiset, 2525
  - std::set, 2683
  - std::tr2::dynamic\_bitset, 2753
  - std::unordered\_map, 2793
  - std::unordered\_multimap, 2812
  - std::unordered\_multiset, 2830
  - std::unordered\_set, 2848
  - std::vector, 2872
- climits, 2963
- clocale, 2963
- clog
  - std, 565
- close
  - \_\_gnu\_cxx::enc\_filebuf, 717
  - \_\_gnu\_cxx::stdio\_filebuf, 767
  - std::basic\_filebuf, 1552
  - std::basic\_fstream, 1580
  - std::basic\_ifstream, 1627
  - std::basic\_ofstream, 1816
- cmath, 2964, 2967
- cmp\_fn
  - \_\_gnu\_pbds::tree, 1223
- cmp\_fn\_imps.hpp, 2970
- code
  - std::regex\_error, 2654
- codecvt.h, 2970
- codecvt\_specializations.h, 2970
- collate
  - std::collate, 2089, 2090
  - std::locale, 2415
  - std::regex\_constants, 614
- combine
  - std::locale, 2412
- common\_type< \_Tp >, 1264
- comp\_ellint\_1
  - Mathematical Special Functions, 243
- comp\_ellint\_2
  - Mathematical Special Functions, 243
- comp\_ellint\_3
  - Mathematical Special Functions, 243
- compare
  - \_\_gnu\_cxx::\_\_versa\_string, 662d
  - \_\_gnu\_debug::basic\_string, 861, 862
  - std::basic\_string, 1945d
  - std::collate, 2090
  - std::collate\_byname, 2094
  - std::sub\_match, 2713, 2714
- Comparison Classes, 217
- compatibility.h, 2971
- compiletime\_settings.h, 2972
  - \_GLIBCXX\_CALL, 2972
- complex, 2973, 2977
  - std::complex, 2098
- Complex Numbers, 28
  - abs, 31
  - acosh, 32
  - arg, 32
  - asinh, 32
  - atanh, 32
  - conj, 32
  - cos, 32
  - cosh, 32
  - exp, 32
  - fabs, 32
  - log, 33
  - log10, 33
  - norm, 33
  - operator<<, 36
  - operator>>, 36
  - operator\*, 33, 34
  - operator\*=: 34
  - operator+, 34
  - operator+=, 34
  - operator-, 34, 35
  - operator-=, 35
  - operator/, 35
  - operator/=: 35
  - operator=, 36
  - operator==, 36
  - polar, 36
  - pow, 37
  - sin, 37
  - sinh, 37
  - sqrt, 37
  - tan, 37

- tanh, [38](#)
- [complex.h](#), [2978](#)
- compose1
  - SGI, [11](#)
- compose2
  - SGI, [11](#)
- [concept\\_check.h](#), [2978](#)
- [concurrency.h](#), [2979](#)
- Concurrency, [22](#)
- [cond\\_dealtor.hpp](#), [2979](#)
- [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#), [2980](#)
- Condition Variables, [39](#)
  - cv\_status, [39](#)
- [condition\\_variable](#), [2980](#)
- conf\_hyperg
  - Mathematical Special Functions, [244](#)
- conj
  - Complex Numbers, [32](#)
- const\_iterator
  - [\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits](#), [1237](#)
  - std::set, [2679](#)
  - std::unordered\_map, [2787](#)
  - std::unordered\_multimap, [2806](#)
  - std::unordered\_multiset, [2825](#)
  - std::unordered\_set, [2842](#)
- [const\\_iterator.hpp](#), [2981](#), [2982](#)
- const\_iterator\_, [1265](#)
  - const\_iterator\_, [1267](#)
  - const\_pointer, [1266](#)
  - const\_reference, [1266](#)
  - const\_iterator\_, [1267](#)
  - difference\_type, [1266](#)
  - iterator\_category, [1266](#)
  - m\_p\_tbl, [1268](#)
  - operator\*, [1267](#)
  - operator++, [1268](#)
  - operator->, [1268](#)
  - operator==, [1268](#)
  - pointer, [1267](#)
  - reference, [1267](#)
  - value\_type, [1267](#)
- const\_local\_iterator
  - std::unordered\_map, [2787](#)
  - std::unordered\_multimap, [2807](#)
  - std::unordered\_multiset, [2825](#)
  - std::unordered\_set, [2843](#)
- const\_pointer
  - [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_](#), [1029](#)
  - [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_](#), [1033](#)
  - [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_](#), [1074](#)
- [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_](#), [1078](#)
- const\_iterator\_, [1266](#)
- iterator\_, [1273](#)
- point\_const\_iterator\_, [1277](#)
- point\_iterator\_, [1279](#)
- std::allocator\_traits, [1496](#)
- std::set, [2679](#)
- std::unordered\_map, [2787](#)
- std::unordered\_multimap, [2807](#)
- std::unordered\_multiset, [2825](#)
- std::unordered\_set, [2843](#)
- const\_pointer\_cast
  - std, [512](#)
- const\_reference
  - [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_iterator\\_](#), [1014](#)
  - [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_iterator\\_](#), [1019](#)
  - [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_](#), [1029](#)
  - [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_](#), [1033](#)
  - [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_](#), [1074](#)
  - [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_](#), [1078](#)
- const\_iterator\_, [1266](#)
- iterator\_, [1273](#)
- point\_const\_iterator\_, [1277](#)
- point\_iterator\_, [1279](#)
- std::set, [2680](#)
- std::unordered\_map, [2787](#)
- std::unordered\_multimap, [2807](#)
- std::unordered\_multiset, [2825](#)
- std::unordered\_set, [2843](#)
- const\_reverse\_iterator
  - std::set, [2680](#)
- const\_void\_pointer
  - [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#), [629](#)
  - std::allocator\_traits, [1496](#)
- constant0
  - SGI, [12](#)
- constant1
  - SGI, [12](#)
- constant2
  - SGI, [12](#)
- construct
  - [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#), [631](#)
  - std::allocator\_traits, [1498](#)
- [constructor\\_destructor\\_fn\\_imps.hpp](#), [2982](#)
- [constructor\\_destructor\\_no\\_store\\_hash\\_fn\\_imps.hpp](#), [2982](#)
- [constructor\\_destructor\\_store\\_hash\\_fn\\_imps.hpp](#), [2982](#), [2983](#)

constructors\_destructor\_fn\_imps.hpp, 2983d  
 container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_Tl >, 1268  
 container\_base\_dispatch.hpp, 2985  
 container\_traits\_base< \_Tag >, 1269  
 container\_type  
     std::back\_insert\_iterator, 1541  
     std::front\_insert\_iterator, 2243  
     std::insert\_iterator, 2300  
 Containers, 16, 248  
 copy  
     \_\_gnu\_cxx::\_\_versa\_string, 664  
     Mutating, 106  
     std::basic\_string, 1947  
 copy\_backward  
     Mutating, 106  
 copy\_exception  
     Exceptions, 25  
 copy\_if  
     Mutating, 106  
 copy\_n  
     Mutating, 107  
     SGI, 12  
 copyfmt  
     std::basic\_fstream, 1580  
     std::basic\_ifstream, 1627  
     std::basic\_ios, 1665  
     std::basic\_iostream, 1690  
     std::basic\_istream, 1736  
     std::basic\_istreamstream, 1777  
     std::basic\_ofstream, 1816  
     std::basic\_ostream, 1848  
     std::basic\_ostreamstream, 1881  
     std::basic\_stringstream, 2002  
 cos  
     Complex Numbers, 32  
 cosh  
     Complex Numbers, 32  
 cout  
     Non-Mutating, 124  
     std, 512  
     std::map, 2436  
     std::multimap, 2509  
     std::multiset, 2525  
     std::set, 2683  
     std::tr2::dynamic\_bitset, 2753  
     std::unordered\_map, 2793  
     std::unordered\_multimap, 2812  
     std::unordered\_multiset, 2830  
     std::unordered\_set, 2848  
 count\_if  
     Non-Mutating, 124  
 count\_minimal\_n  
     \_\_gnu\_parallel::\_Settings, 961  
 cout  
     std, 565  
 cpp\_type\_traits.h, 2986  
 cpu\_defines.h, 2986  
 crbegin  
     \_\_gnu\_cxx::\_\_versa\_string, 665  
     std::\_\_detail::\_Nfa, 1398  
     std::basic\_string, 1948  
     std::deque, 2186  
     std::list, 2398  
     std::map, 2436  
     std::multimap, 2510  
     std::multiset, 2525  
     std::set, 2684  
     std::vector, 2873  
 cref  
     std, 512  
 cregex\_token\_iterator  
     Regular Expressions, 187  
 crend  
     \_\_gnu\_cxx::\_\_versa\_string, 665  
     std::\_\_detail::\_Nfa, 1398  
     std::basic\_string, 1948  
     std::deque, 2186  
     std::list, 2398  
     std::map, 2436  
     std::multimap, 2510  
     std::multiset, 2525  
     std::set, 2684  
     std::vector, 2873  
 csetjmp, 2986  
 cshift  
     Numeric Arrays, 87  
 csignal, 2987  
 cstdarg, 2987, 2988  
 cstdbool, 2988  
 cstdddef, 2988  
 cstdint, 2989  
 cstdio, 2989, 2990  
 cstdlib, 2990, 2991  
 cstring, 2991  
 csub\_match  
     Regular Expressions, 187  
 ctgmath, 2991, 2992  
 ctime, 2992  
 ctype  
     std::ctype< char >, 2122  
     std::ctype< wchar\_t >, 2133  
     std::locale, 2415  
 ctype\_base.h, 2992  
 ctype\_inline.h, 2993  
 cur  
     std::basic\_fstream, 1614  
     std::basic\_ifstream, 1654

- std::basic\_ios, 1677
- std::basic\_iostream, 1724
- std::basic\_istream, 1762
- std::basic\_istream, 1803
- std::basic\_ofstream, 1836
- std::basic\_ostream, 1867
- std::basic\_ostringstream, 1900
- std::basic\_stringstream, 2036
- std::ios\_base, 2315
- curr\_symbol
  - std::moneypunct, 2489
  - std::moneypunct\_byname, 2497
- current\_exception
  - Exceptions, 25
- cv\_status
  - Condition Variables, 39
- cwchar, 2993, 2994
- cwctype, 2994
- cxxabi.h, 2995
- cxxabi\_forced.h, 2996
- cxxabi\_tweaks.h, 2996
- cyl\_bessel\_i
  - Mathematical Special Functions, 244
- cyl\_bessel\_j
  - Mathematical Special Functions, 244
- cyl\_bessel\_k
  - Mathematical Special Functions, 244
- cyl\_neumann
  - Mathematical Special Functions, 244
- data
  - \_\_gnu\_cxx::\_\_versa\_string, 665
  - std::\_\_detail::\_\_Nfa, 1398
  - std::basic\_string, 1948
  - std::vector, 2873
- Data Structure Type, 261
- date\_order
  - std::time\_get, 2721
  - std::time\_get\_byname, 2729
- deallocate
  - \_\_gnu\_cxx::\_\_alloc\_traits, 631
  - std::allocator\_traits, 1498
- debug.h, 2997
- debug\_allocator.h, 2998
- debug\_fn\_imps.hpp, 2998d
- debug\_map\_base.hpp, 3000
- debug\_no\_store\_hash\_fn\_imps.hpp, 3001
- debug\_store\_hash\_fn\_imps.hpp, 3001
- dec
  - std, 512
  - std::basic\_fstream, 1614
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1677
  - std::basic\_iostream, 1724
  - std::basic\_istream, 1762
  - std::basic\_istream, 1803
  - std::basic\_ofstream, 1836
  - std::basic\_ostream, 1867
  - std::basic\_ostringstream, 1900
  - std::basic\_stringstream, 2036
  - std::ios\_base, 2315
- decimal, 3001
- Decimal Floating-Point Arithmetic, 247
- decimal128
  - std::decimal::decimal128, 2168
- decimal32
  - std::decimal::decimal32, 2169
- decimal32\_to\_long\_long
  - std::decimal, 609
- decimal64
  - std::decimal::decimal64, 2171
- decimal\_point
  - std::moneypunct, 2489
  - std::moneypunct\_byname, 2497
  - std::numpunct, 2597
  - std::numpunct\_byname, 2602
- declval
  - Utilities, 67
- default\_trie\_access\_traits< Key >, 1269
- denorm\_absent
  - std, 497
- denorm\_indeterminate
  - std, 497
- denorm\_present
  - std, 497
- denorm\_min
  - std::numeric\_limits, 2572
- densities
  - std::piecewise\_constant\_distribution, 2620
  - std::piecewise\_linear\_distribution, 2625
- deque, 3011, 3012
  - std::deque, 2177d
- deque.tcc, 3012
- destroy
  - \_\_gnu\_cxx::\_\_alloc\_traits, 631
  - std::allocator\_traits, 1499
- Diagnostics, 21
- difference\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator\_, 1014
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator\_, 1019
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 1029
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 1033
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 1074

- `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_`, 1078
- `const_iterator_`, 1266
- `iterator_`, 1273
- `point_const_iterator_`, 1277
- `point_iterator_`, 1279
- `std::allocator_traits`, 1496
- `std::back_insert_iterator`, 1541
- `std::front_insert_iterator`, 2243
- `std::insert_iterator`, 2300
- `std::istream_iterator`, 2374
- `std::istreambuf_iterator`, 2377
- `std::iterator`, 2381
- `std::ostream_iterator`, 2607
- `std::ostreambuf_iterator`, 2610
- `std::pointer_traits`, 2632
- `std::pointer_traits< _Tp * >`, 2633
- `std::raw_storage_iterator`, 2651
- `std::set`, 2680
- `std::unordered_map`, 2787
- `std::unordered_multimap`, 2807
- `std::unordered_multiset`, 2825
- `std::unordered_set`, 2843
- `digits`
  - `std::__numeric_limits_base`, 1421
  - `std::numeric_limits`, 2573
- `digits10`
  - `std::__numeric_limits_base`, 1421
  - `std::numeric_limits`, 2573
- `direct_mask_range_hashing_imp.hpp`, 3013
- `direct_mod_range_hashing_imp.hpp`, 3013
- `discard`
  - `std::discard_block_engine`, 2196
  - `std::independent_bits_engine`, 2294
  - `std::linear_congruential_engine`, 2388
  - `std::mersenne_twister_engine`, 2466
  - `std::shuffle_order_engine`, 2701
- `discard_block_engine`
  - `std::discard_block_engine`, 2195, 2196
- `distance`
  - `SGL`, 12
  - `std`, 512
- `do_compare`
  - `std::collate`, 2090
  - `std::collate_byname`, 2095
- `do_curr_symbol`
  - `std::moneypunct`, 2489
  - `std::moneypunct_byname`, 2497
- `do_date_order`
  - `std::time_get`, 2721
  - `std::time_get_byname`, 2729
- `do_decimal_point`
  - `std::moneypunct`, 2490
  - `std::moneypunct_byname`, 2498
- `std::numpunct`, 2598
- `std::numpunct_byname`, 2602
- `do_falsename`
  - `std::numpunct`, 2598
  - `std::numpunct_byname`, 2602
- `do_frac_digits`
  - `std::moneypunct`, 2490
  - `std::moneypunct_byname`, 2498
- `do_get`
  - `std::money_get`, 2479, 2480
  - `std::num_get`, 2549d
- `do_get_date`
  - `std::time_get`, 2722
  - `std::time_get_byname`, 2729
- `do_get_monthname`
  - `std::time_get`, 2722
  - `std::time_get_byname`, 2730
- `do_get_time`
  - `std::time_get`, 2723
  - `std::time_get_byname`, 2731
- `do_get_weekday`
  - `std::time_get`, 2723
  - `std::time_get_byname`, 2731
- `do_get_year`
  - `std::time_get`, 2724
  - `std::time_get_byname`, 2732
- `do_grouping`
  - `std::moneypunct`, 2490
  - `std::moneypunct_byname`, 2498
  - `std::numpunct`, 2598
  - `std::numpunct_byname`, 2603
- `do_hash`
  - `std::collate`, 2091
  - `std::collate_byname`, 2095
- `do_is`
  - `std::__ctype_abstract_base`, 1293
  - `std::ctype`, 2110
  - `std::ctype< wchar_t >`, 2134
  - `std::ctype_byname`, 2146
- `do_narrow`
  - `std::__ctype_abstract_base`, 1294
  - `std::ctype`, 2111
  - `std::ctype< char >`, 2123
  - `std::ctype< wchar_t >`, 2134, 2135
  - `std::ctype_byname`, 2147
  - `std::ctype_byname< char >`, 2158
- `do_neg_format`
  - `std::moneypunct`, 2491
  - `std::moneypunct_byname`, 2499
- `do_negative_sign`
  - `std::moneypunct`, 2491
  - `std::moneypunct_byname`, 2499
- `do_out`
  - `std::__codecvt_abstract_base`, 1288

- std::codecvt, 2069
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2073
- std::codecvt< char, char, mbstate\_t >, 2077
- std::codecvt< wchar\_t, char, mbstate\_t >, 2080
- std::codecvt\_byname, 2085
- do\_pos\_format
  - std::moneypunct, 2491
  - std::moneypunct\_byname, 2499
- do\_positive\_sign
  - std::moneypunct, 2492
  - std::moneypunct\_byname, 2500
- do\_put
  - std::money\_put, 2484
  - std::num\_put, 2563d
  - std::time\_put, 2737
  - std::time\_put\_byname, 2740
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base, 1294
  - std::ctype, 2112
  - std::ctype< wchar\_t >, 2135
  - std::ctype\_byname, 2147
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base, 1295
  - std::ctype, 2112
  - std::ctype< wchar\_t >, 2136
  - std::ctype\_byname, 2148
- do\_thousands\_sep
  - std::moneypunct, 2492
  - std::moneypunct\_byname, 2500
  - std::numpunct, 2598
  - std::numpunct\_byname, 2603
- do\_tolower
  - std::\_\_ctype\_abstract\_base, 1295, 1296
  - std::ctype, 2113
  - std::ctype< char >, 2123, 2124
  - std::ctype< wchar\_t >, 2136
  - std::ctype\_byname, 2148, 2149
  - std::ctype\_byname< char >, 2159
- do\_toupper
  - std::\_\_ctype\_abstract\_base, 1296
  - std::ctype, 2113, 2114
  - std::ctype< char >, 2124
  - std::ctype< wchar\_t >, 2137
  - std::ctype\_byname, 2149
  - std::ctype\_byname< char >, 2159, 2160
- do\_transform
  - std::collate, 2091
  - std::collate\_byname, 2096
- do\_truename
  - std::numpunct, 2599
  - std::numpunct\_byname, 2603
- do\_widen
  - std::\_\_ctype\_abstract\_base, 1297
- std::ctype, 2114, 2115
- std::ctype< char >, 2125
- std::ctype< wchar\_t >, 2137, 2138
- std::ctype\_byname, 2150
- std::ctype\_byname< char >, 2160
- duration\_cast
  - std::chrono, 600
- dynamic\_bitset, 3014
  - std::tr2::dynamic\_bitset, 2751d
- dynamic\_pointer\_cast
  - std, 513
- e\_pos
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1216
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 1237
- e\_type
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1216
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 1237
- ECMAScript
  - std::regex\_constants, 614
- eback
  - \_\_gnu\_cxx::enc\_filebuf, 717
  - \_\_gnu\_cxx::stdio\_filebuf, 767
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 787
  - std::basic\_filebuf, 1553
  - std::basic\_streambuf, 1916
  - std::basic\_stringbuf, 1978
- egptr
  - \_\_gnu\_cxx::enc\_filebuf, 717
  - \_\_gnu\_cxx::stdio\_filebuf, 768
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 787
  - std::basic\_filebuf, 1553
  - std::basic\_streambuf, 1917
  - std::basic\_stringbuf, 1978
- egrep
  - std::regex\_constants, 614
- element\_type
  - std::auto\_ptr, 1536
  - std::pointer\_traits, 2632
  - std::pointer\_traits< \_Tp \* >, 2633
- ellint\_1
  - Mathematical Special Functions, 244
- ellint\_2
  - Mathematical Special Functions, 244
- ellint\_3
  - Mathematical Special Functions, 244
- emplace
  - std::\_\_detail::\_\_Nfa, 1398
  - std::deque, 2186
  - std::list, 2399
  - std::map, 2437
  - std::multimap, 2510
  - std::multiset, 2525
  - std::set, 2684

- std::unordered\_map, 2793
- std::unordered\_multimap, 2813
- std::unordered\_multiset, 2830
- std::unordered\_set, 2848
- std::vector, 2873
- emplace\_after
  - std::forward\_list, 2231
- emplace\_front
  - std::forward\_list, 2231
- emplace\_hint
  - std::map, 2437
  - std::multimap, 2510
  - std::multiset, 2526
  - std::set, 2684
  - std::unordered\_map, 2794
  - std::unordered\_multimap, 2813
  - std::unordered\_multiset, 2831
  - std::unordered\_set, 2848
- empty
  - \_\_gnu\_cxx::\_\_versa\_string, 665
  - \_\_gnu\_debug::basic\_string, 862
  - \_\_gnu\_pbds::detail::cc\_ht\_map, 1044
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 1067
  - std::\_\_detail::Nfa, 1399
  - std::basic\_string, 1948
  - std::deque, 2186
  - std::forward\_list, 2232
  - std::list, 2399
  - std::map, 2437
  - std::match\_results, 2453
  - std::multimap, 2511
  - std::multiset, 2526
  - std::priority\_queue, 2639
  - std::queue, 2642
  - std::set, 2685
  - std::stack, 2707
  - std::tr2::dynamic\_bitset, 2754
  - std::unordered\_map, 2794
  - std::unordered\_multimap, 2813
  - std::unordered\_multiset, 2831
  - std::unordered\_set, 2849
  - std::vector, 2873
- enc\_filebuf.h, 3015
- end
  - \_\_gnu\_cxx::\_\_versa\_string, 666
  - \_\_gnu\_cxx::temporary\_buffer, 803
  - \_\_gnu\_parallel::\_PseudoSequence, 952
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1216
  - \_\_gnu\_pbds::trie\_string\_access\_traits, 1238
  - Numeric Arrays, 87
  - std, 513, 514
  - std::\_\_detail::Nfa, 1399
  - std::\_Temporary\_buffer, 1480
  - std::basic\_fstream, 1614
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1678
  - std::basic\_iostream, 1724
  - std::basic\_istream, 1762
  - std::basic\_istreamstream, 1803
  - std::basic\_ofstream, 1836
  - std::basic\_ostream, 1867
  - std::basic\_ostreamstream, 1900
  - std::basic\_string, 1948
  - std::basic\_stringstream, 2036
  - std::deque, 2186
  - std::forward\_list, 2232
  - std::ios\_base, 2315
  - std::list, 2399
  - std::map, 2438
  - std::match\_results, 2454
  - std::multimap, 2511
  - std::multiset, 2526
  - std::set, 2685
  - std::unordered\_map, 2794, 2795
  - std::unordered\_multimap, 2814
  - std::unordered\_multiset, 2831, 2832
  - std::unordered\_set, 2849, 2850
  - std::vector, 2874
- endl
  - std, 514
- ends
  - std, 514
- entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, No\_Throw >, 1270
- entry\_cmp.hpp, 3015
- entry\_list\_fn\_imps.hpp, 3016
- entry\_metadata\_base.hpp, 3016
- entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw >, 1270
- entry\_pred.hpp, 3016
- eof
  - std::basic\_fstream, 1581
  - std::basic\_ifstream, 1627
  - std::basic\_ios, 1666
  - std::basic\_iostream, 1691
  - std::basic\_istream, 1736
  - std::basic\_istreamstream, 1777
  - std::basic\_ofstream, 1816
  - std::basic\_ostream, 1848
  - std::basic\_ostreamstream, 1881
  - std::basic\_stringstream, 2003
- eofbit
  - std::basic\_fstream, 1614
  - std::basic\_ifstream, 1654
  - std::basic\_ios, 1678
  - std::basic\_iostream, 1724
  - std::basic\_istream, 1763
  - std::basic\_istreamstream, 1803
  - std::basic\_ofstream, 1836
  - std::basic\_ostream, 1867

- `std::basic_ostringstream`, 1901
  - `std::basic_stringstream`, 2036
  - `std::ios_base`, 2315
- `ep_ptr`
  - `__gnu_cxx::enc_filebuf`, 717
  - `__gnu_cxx::stdio_filebuf`, 768
  - `__gnu_cxx::stdio_sync_filebuf`, 787
  - `std::basic_filebuf`, 1553
  - `std::basic_streambuf`, 1917
  - `std::basic_stringbuf`, 1978
- `epsilon`
  - `std::numeric_limits`, 2572
- `eq_by_less.hpp`, 3017
- `equal`
  - Non-Mutating, 124, 125
  - `std::istreambuf_iterator`, 2378
- `equal_range`
  - Binary Search, 157, 158
  - `std::map`, 2438
  - `std::multimap`, 2511, 2512
  - `std::multiset`, 2527
  - `std::set`, 2685
  - `std::unordered_map`, 2795
  - `std::unordered_multimap`, 2814, 2815
  - `std::unordered_multiset`, 2832
  - `std::unordered_set`, 2850
- `equally_split.h`, 3017
- `equals`
  - `std::tr2::bool_set`, 2746
- `erase`
  - `__gnu_cxx::__versa_string`, 666, 667
  - `__gnu_debug::basic_string`, 862, 863
  - `std::__detail::__Nfa`, 1399
  - `std::basic_string`, 1949
  - `std::deque`, 2187
  - `std::list`, 2399, 2400
  - `std::map`, 2439, 2440
  - `std::multimap`, 2512, 2513
  - `std::multiset`, 2527, 2528
  - `std::set`, 2686
  - `std::unordered_map`, 2796, 2797
  - `std::unordered_multimap`, 2815, 2816
  - `std::unordered_multiset`, 2833, 2834
  - `std::unordered_set`, 2851, 2852
  - `std::vector`, 2874
- `erase_can_throw`
  - `__gnu_pbds::container_traits`, 1005
- `erase_after`
  - `std::forward_list`, 2232
- `erase_fn_imps.hpp`, 3017d
- `erase_no_store_hash_fn_imps.hpp`, 3020
- `erase_store_hash_fn_imps.hpp`, 3020
- `error_backref`
  - `std::regex_constants`, 612
- `error_badbrace`
  - `std::regex_constants`, 612
- `error_badrepeat`
  - `std::regex_constants`, 613
- `error_brace`
  - `std::regex_constants`, 613
- `error_brack`
  - `std::regex_constants`, 613
- `error_collate`
  - `std::regex_constants`, 613
- `error_complexity`
  - `std::regex_constants`, 613
- `error_constants.h`, 3020
- `error_ctype`
  - `std::regex_constants`, 613
- `error_escape`
  - `std::regex_constants`, 613
- `error_paren`
  - `std::regex_constants`, 613
- `error_range`
  - `std::regex_constants`, 613
- `error_space`
  - `std::regex_constants`, 613
- `error_stack`
  - `std::regex_constants`, 613
- `error_type`
  - `std::regex_constants`, 612
- `event`
  - `std::basic_fstream`, 1578
  - `std::basic_ifstream`, 1625
  - `std::basic_ios`, 1664
  - `std::basic_iostream`, 1689
  - `std::basic_istream`, 1734
  - `std::basic_istreamstream`, 1775
  - `std::basic_ofstream`, 1813
  - `std::basic_ostream`, 1846
  - `std::basic_ostringstream`, 1879
  - `std::basic_stringstream`, 2000
  - `std::ios_base`, 2308
- `event_callback`
  - `std::basic_fstream`, 1576
  - `std::basic_ifstream`, 1623
  - `std::basic_ios`, 1662
  - `std::basic_iostream`, 1687
  - `std::basic_istream`, 1732
  - `std::basic_istreamstream`, 1773
  - `std::basic_ofstream`, 1812
  - `std::basic_ostream`, 1844
  - `std::basic_ostringstream`, 1877
  - `std::basic_stringstream`, 1998
  - `std::ios_base`, 2307
- `exception`, 3021
- `exception.hpp`, 3021
- `exception_defines.h`, 3022

- exception\_ptr.h, [3022](#)
- Exceptions, [23](#), [254](#)
  - \_\_verbose\_terminate\_handler, [25](#)
  - copy\_exception, [25](#)
  - current\_exception, [25](#)
  - make\_exception\_ptr, [25](#)
  - rethrow\_exception, [25](#)
  - rethrow\_if\_nested, [25](#), [26](#)
  - set\_terminate, [26](#)
  - set\_unexpected, [26](#)
  - terminate, [26](#)
  - terminate\_handler, [25](#)
  - throw\_with\_nested, [26](#)
  - uncaught\_exception, [26](#)
  - unexpected, [26](#)
  - unexpected\_handler, [25](#)
- exceptions
  - std::basic\_fstream, [1581](#)
  - std::basic\_ifstream, [1628](#)
  - std::basic\_ios, [1666](#)
  - std::basic\_iostream, [1691](#)
  - std::basic\_istream, [1736](#), [1737](#)
  - std::basic\_istreamstream, [1777](#)
  - std::basic\_ofstream, [1816](#), [1817](#)
  - std::basic\_ostream, [1848](#), [1849](#)
  - std::basic\_ostringstream, [1881](#), [1882](#)
  - std::basic\_stringstream, [2003](#)
- exp
  - Complex Numbers, [32](#)
- expint
  - Mathematical Special Functions, [245](#)
- exponential\_distribution
  - std::exponential\_distribution, [2212](#)
- extc++.h, [3023](#)
- extended
  - std::regex\_constants, [614](#)
- Extensions, [5](#)
- external\_load\_access
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [994](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, [1181](#)
- extptr\_allocator.h, [3023](#)
- fabs
  - Complex Numbers, [32](#)
  - std, [514](#)
- facet
  - std::locale::facet, [2418](#)
- fail
  - std::basic\_fstream, [1582](#)
  - std::basic\_ifstream, [1628](#)
  - std::basic\_ios, [1667](#)
  - std::basic\_iostream, [1692](#)
  - std::basic\_istream, [1737](#)
- std::basic\_istreamstream, [1778](#)
- std::basic\_ofstream, [1817](#)
- std::basic\_ostream, [1849](#)
- std::basic\_ostringstream, [1882](#)
- std::basic\_stringstream, [2004](#)
- failbit
  - std::basic\_fstream, [1615](#)
  - std::basic\_ifstream, [1655](#)
  - std::basic\_ios, [1678](#)
  - std::basic\_iostream, [1725](#)
  - std::basic\_istream, [1763](#)
  - std::basic\_istreamstream, [1804](#)
  - std::basic\_ofstream, [1837](#)
  - std::basic\_ostream, [1868](#)
  - std::basic\_ostringstream, [1901](#)
  - std::basic\_stringstream, [2037](#)
  - std::ios\_base, [2315](#)
- failed
  - std::ostreambuf\_iterator, [2612](#)
- false\_type
  - Metaprogramming, [63](#)
- falsename
  - std::numpunct, [2599](#)
  - std::numpunct\_byname, [2603](#)
- fd
  - \_\_gnu\_cxx::stdio\_filebuf, [768](#)
- features.h, [3024](#)
  - \_GLIBCXX\_MERGESORT, [3025](#)
  - \_GLIBCXX\_QUICKSORT, [3025](#)
- fenv.h, [3026](#)
- file
  - \_\_gnu\_cxx::stdio\_filebuf, [769](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [788](#)
- filebuf
  - I/O, [43](#)
- fill
  - Mutating, [107](#)
  - std::basic\_fstream, [1582](#)
  - std::basic\_ifstream, [1629](#)
  - std::basic\_ios, [1667](#)
  - std::basic\_iostream, [1692](#)
  - std::basic\_istream, [1737](#), [1738](#)
  - std::basic\_istreamstream, [1778](#)
  - std::basic\_ofstream, [1817](#), [1818](#)
  - std::basic\_ostream, [1850](#)
  - std::basic\_ostringstream, [1883](#)
  - std::basic\_stringstream, [2004](#)
- fill\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [961](#)
- fill\_n
  - Mutating, [108](#)
- find
  - \_\_gnu\_cxx::\_\_versa\_string, [667](#), [668](#)
  - \_\_gnu\_debug::basic\_string, [863](#)

- Non-Mutating, [125](#)
- std::basic\_string, [1950](#), [1951](#)
- std::map, [2440](#)
- std::multimap, [2513](#)
- std::multiset, [2529](#)
- std::set, [2687](#)
- std::unordered\_map, [2797](#), [2798](#)
- std::unordered\_multimap, [2816](#), [2817](#)
- std::unordered\_multiset, [2834](#), [2835](#)
- std::unordered\_set, [2852](#), [2853](#)
- find.h, [3026](#)
- find\_by\_order
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update, [1226](#)
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, [1232](#)
- find\_end
  - Non-Mutating, [125](#), [126](#)
- find\_first
  - std::tr2::dynamic\_bitset, [2754](#)
- find\_first\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, [669](#), [670](#)
  - \_\_gnu\_debug::basic\_string, [863](#)
  - std::basic\_string, [1951](#), [1952](#)
- find\_first\_of
  - \_\_gnu\_cxx::\_\_versa\_string, [671](#), [672](#)
  - \_\_gnu\_debug::basic\_string, [864](#)
  - Non-Mutating, [126](#), [127](#)
  - std::basic\_string, [1953](#), [1954](#)
- find\_fn\_imps.hpp, [3026d](#)
- find\_if
  - Non-Mutating, [127](#)
- find\_if\_not
  - Non-Mutating, [127](#)
- find\_increasing\_factor
  - \_\_gnu\_parallel::Settings, [961](#)
- find\_initial\_block\_size
  - \_\_gnu\_parallel::Settings, [961](#)
- find\_last\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, [672](#), [673](#)
  - \_\_gnu\_debug::basic\_string, [864](#)
  - std::basic\_string, [1954](#), [1955](#)
- find\_last\_of
  - \_\_gnu\_cxx::\_\_versa\_string, [674](#), [675](#)
  - \_\_gnu\_debug::basic\_string, [864](#)
  - std::basic\_string, [1956](#), [1957](#)
- find\_maximum\_block\_size
  - \_\_gnu\_parallel::Settings, [962](#)
- find\_next
  - std::tr2::dynamic\_bitset, [2754](#)
- find\_no\_store\_hash\_fn\_imps.hpp, [3028](#)
- find\_scale\_factor
  - \_\_gnu\_parallel::Settings, [962](#)
- find\_selectors.h, [3028](#)
- find\_sequential\_search\_size
  - \_\_gnu\_parallel::Settings, [962](#)
- find\_store\_hash\_fn\_imps.hpp, [3029](#)
- first
  - \_\_gnu\_parallel::IteratorPair, [919](#)
  - std::pair, [2619](#)
  - std::sub\_match, [2715](#)
- first\_argument\_type
  - \_\_gnu\_cxx::project1st, [746](#)
  - \_\_gnu\_cxx::project2nd, [747](#)
  - \_\_gnu\_parallel::EqualFromLess, [913](#)
  - \_\_gnu\_parallel::EqualTo, [915](#)
  - \_\_gnu\_parallel::Less, [922](#)
  - \_\_gnu\_parallel::Lexicographic, [924](#)
  - \_\_gnu\_parallel::LexicographicReverse, [925](#)
  - \_\_gnu\_parallel::Multiplies, [947](#)
  - \_\_gnu\_parallel::Plus, [949](#)
- std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 >, [1469](#)
- std::binary\_function, [2045](#)
- std::binary\_negate, [2046](#)
- std::const\_mem\_fun1\_ref\_t, [2103](#)
- std::const\_mem\_fun1\_t, [2105](#)
- std::divides, [2203](#)
- std::equal\_to, [2207](#)
- std::greater, [2260](#)
- std::greater\_equal, [2262](#)
- std::less, [2384](#)
- std::less\_equal, [2386](#)
- std::logical\_and, [2423](#)
- std::logical\_or, [2425](#)
- std::mem\_fun1\_ref\_t, [2459](#)
- std::mem\_fun1\_t, [2460](#)
- std::minus, [2474](#)
- std::modulus, [2476](#)
- std::multiplies, [2520](#)
- std::not\_equal\_to, [2546](#)
- std::owner\_less< shared\_ptr< \_Tp > >, [2615](#)
- std::owner\_less< weak\_ptr< \_Tp > >, [2616](#)
- std::plus, [2628](#)
- std::pointer\_to\_binary\_function, [2630](#)
- fixed
  - std, [514](#)
  - std::basic\_fstream, [1615](#)
  - std::basic\_ifstream, [1655](#)
  - std::basic\_ios, [1678](#)
  - std::basic\_iostream, [1725](#)
  - std::basic\_istream, [1763](#)
  - std::basic\_istreamstream, [1804](#)
  - std::basic\_ofstream, [1837](#)
  - std::basic\_ostream, [1868](#)
  - std::basic\_ostreamstream, [1901](#)
  - std::basic\_stringstream, [2037](#)
  - std::ios\_base, [2316](#)

- flags
  - std::basic\_fstream, 1583
  - std::basic\_ifstream, 1629
  - std::basic\_ios, 1668
  - std::basic\_iostream, 1693
  - std::basic\_istream, 1738
  - std::basic\_istreamstream, 1779
  - std::basic\_ofstream, 1818
  - std::basic\_ostream, 1850
  - std::basic\_ostreamstream, 1883
  - std::basic\_regex, 1910
  - std::basic\_stringstream, 2005
  - std::ios\_base, 2309
- flip
  - std, 515
  - std::tr2::dynamic\_bitset, 2754
- float\_denorm\_style
  - std, 497
- float\_round\_style
  - std, 498
- floatfield
  - std::basic\_fstream, 1615
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1678
  - std::basic\_iostream, 1725
  - std::basic\_istream, 1763
  - std::basic\_istreamstream, 1804
  - std::basic\_ofstream, 1837
  - std::basic\_ostream, 1868
  - std::basic\_ostreamstream, 1901
  - std::basic\_stringstream, 2037
  - std::ios\_base, 2316
- flush
  - std, 515
  - std::basic\_fstream, 1583
  - std::basic\_iostream, 1693
  - std::basic\_ofstream, 1819
  - std::basic\_ostream, 1851
  - std::basic\_ostreamstream, 1884
  - std::basic\_stringstream, 2005
- fmtflags
  - std::basic\_fstream, 1576
  - std::basic\_ifstream, 1623
  - std::basic\_ios, 1662
  - std::basic\_iostream, 1687
  - std::basic\_istream, 1733
  - std::basic\_istreamstream, 1773
  - std::basic\_ofstream, 1812
  - std::basic\_ostream, 1845
  - std::basic\_ostreamstream, 1877
  - std::basic\_stringstream, 1999
  - std::ios\_base, 2307
- for\_each
  - Non-Mutating, 128
- for\_each.h, 3029
- for\_each\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 962
- for\_each\_selectors.h, 3030
- format
  - std::match\_results, 2454
- format\_default
  - std::regex\_constants, 614
- format\_first\_only
  - std::regex\_constants, 615
- format\_no\_copy
  - std::regex\_constants, 615
- format\_sed
  - std::regex\_constants, 615
- formatter.h, 3031
- forward
  - Utilities, 68
- forward\_list, 3031d
  - std::forward\_list, 2227, 2228
- forward\_list.h, 3033
- forward\_list.tcc, 3034
- fpos
  - std::fpos, 2241
- frac\_digits
  - std::moneypunct, 2492
  - std::moneypunct\_byname, 2500
- front
  - \_\_gnu\_cxx::\_\_versa\_string, 675, 676
  - \_\_gnu\_debug::basic\_string, 865
  - std::\_\_detail::\_Nfa, 1400
  - std::basic\_string, 1957
  - std::deque, 2187
  - std::forward\_list, 2233
  - std::list, 2400
  - std::queue, 2642, 2643
  - std::vector, 2875
- front\_insert\_iterator
  - std::front\_insert\_iterator, 2244
- front\_inserter
  - Iterators, 233
- fstream, 3035
  - I/O, 43
- fstream.tcc, 3035
- functexcept.h, 3036
- function
  - std::function< \_Res(\_ArgTypes...)>, 2246, 2247
- Function Objects, 214
  - mem\_fn, 215
- functional, 3037, 3041
- functional\_hash.h, 3042
- functions.h, 3044
- future, 3046
- future\_category
  - Futures, 41

- future\_errc
  - Futures, 41
- future\_status
  - Futures, 41
- Futures, 40
  - future\_category, 41
  - future\_errc, 41
  - future\_status, 41
  - launch, 41
  - make\_error\_code, 41
  - make\_error\_condition, 41
- gamma\_distribution
  - std::gamma\_distribution, 2253
- gbump
  - \_\_gnu\_cxx::enc\_filebuf, 718
  - \_\_gnu\_cxx::stdio\_filebuf, 769
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 788
  - std::basic\_filebuf, 1554
  - std::basic\_streambuf, 1917
  - std::basic\_stringbuf, 1979
- gcount
  - std::basic\_fstream, 1583
  - std::basic\_ifstream, 1630
  - std::basic\_iostream, 1693
  - std::basic\_istream, 1739
  - std::basic\_istreamstream, 1779
  - std::basic\_stringstream, 2005
- generate
  - Mutating, 108
- generate\_canonical
  - Random Number Generation, 181
- generate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 962
- generate\_n
  - Mutating, 108
- get
  - \_\_gnu\_parallel::Settings, 961
  - std::auto\_ptr, 1537
  - std::basic\_fstream, 1584d
  - std::basic\_ifstream, 1630d
  - std::basic\_iostream, 1694d
  - std::basic\_istream, 1739d
  - std::basic\_istreamstream, 1779d
  - std::basic\_stringstream, 2006d
  - std::money\_get, 2480
  - std::num\_get, 2554d
- get\_actual\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, 1185
- get\_allocator
  - \_\_gnu\_cxx::\_\_versa\_string, 676
  - \_\_gnu\_debug::basic\_string, 865
  - std::basic\_string, 1957
  - std::deque, 2188
  - std::forward\_list, 2233
  - std::list, 2400
  - std::map, 2441
  - std::match\_results, 2455
  - std::multimap, 2514
  - std::multiset, 2529
  - std::set, 2687
  - std::tr2::dynamic\_bitset, 2755
  - std::unordered\_map, 2798
  - std::unordered\_multimap, 2817
  - std::unordered\_multiset, 2835
  - std::unordered\_set, 2853
- get\_child
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 1113
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 1116
- get\_comb\_hash\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map, 1044
- get\_comb\_probe\_fn
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 1067
- get\_date
  - std::time\_get, 2724
  - std::time\_get\_byname, 2732
- get\_deleter
  - Pointer Abstractions, 51
- get\_eq\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map, 1044, 1045
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 1067
- get\_hash\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map, 1045
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 1067, 1068
- get\_id
  - std::this\_thread, 618
- get\_l\_child
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, 1015
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 1020
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_, 1092
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_, 1094
- get\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, 995
- get\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 1182
- get\_metadata
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, 1015
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 1020
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 1113
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 1116
- get\_money
  - std, 515
- get\_monthname

- std::time\_get, 2725
- std::time\_get\_byname, 2733
- get\_nearest\_larger\_size
  - \_\_gnu\_pbds::sample\_size\_policy, 1215
- get\_nearest\_smaller\_size
  - \_\_gnu\_pbds::sample\_size\_policy, 1215
- get\_new\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, 1185
  - \_\_gnu\_pbds::sample\_resize\_policy, 1210
- get\_probe\_fn
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 1068
- get\_r\_child
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, 1015
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 1020
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_, 1092
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_, 1094
- get\_resize\_policy
  - \_\_gnu\_pbds::detail::cc\_ht\_map, 1045
  - \_\_gnu\_pbds::detail::gp\_ht\_map, 1068
- get\_size\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, 1185
- get\_temporary\_buffer
  - std, 515
- get\_time
  - std::time\_get, 2725
  - std::time\_get\_byname, 2733
- get\_trigger\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, 1185
- get\_weekday
  - std::time\_get, 2726
  - std::time\_get\_byname, 2734
- get\_year
  - std::time\_get, 2726
  - std::time\_get\_byname, 2734
- getline
  - std, 516, 517
  - std::basic\_fstream, 1586, 1587
  - std::basic\_ifstream, 1632, 1633
  - std::basic\_iostream, 1696, 1697
  - std::basic\_istream, 1741, 1742
  - std::basic\_istreamstream, 1782, 1783
  - std::basic\_stringstream, 2008, 2009
- getloc
  - \_\_gnu\_cxx::enc\_filebuf, 718
  - \_\_gnu\_cxx::stdio\_filebuf, 769
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 788
  - std::basic\_filebuf, 1554
  - std::basic\_fstream, 1587
  - std::basic\_ifstream, 1633
  - std::basic\_ios, 1668
  - std::basic\_iostream, 1697
  - std::basic\_istream, 1742
  - std::basic\_istreamstream, 1783
- std::basic\_ofstream, 1819
- std::basic\_ostream, 1851
- std::basic\_ostringstream, 1884
- std::basic\_regex, 1910
- std::basic\_streambuf, 1918
- std::basic\_stringbuf, 1979
- std::basic\_stringstream, 2009
- std::ios\_base, 2309
- std::regex\_traits, 2662
- global
  - std::locale, 2412
- good
  - std::basic\_fstream, 1587
  - std::basic\_ifstream, 1634
  - std::basic\_ios, 1668
  - std::basic\_iostream, 1697
  - std::basic\_istream, 1743
  - std::basic\_istreamstream, 1783
  - std::basic\_ofstream, 1819
  - std::basic\_ostream, 1851
  - std::basic\_ostringstream, 1884
  - std::basic\_stringstream, 2009
- goodbit
  - std::basic\_fstream, 1615
  - std::basic\_ifstream, 1655
  - std::basic\_ios, 1679
  - std::basic\_iostream, 1725
  - std::basic\_istream, 1763
  - std::basic\_istreamstream, 1804
  - std::basic\_ofstream, 1837
  - std::basic\_ostream, 1868
  - std::basic\_ostringstream, 1901
  - std::basic\_stringstream, 2037
  - std::ios\_base, 2316
- gp\_hash\_table
  - \_\_gnu\_pbds::gp\_hash\_table, 1176d
- gp\_ht\_map.hpp, 3047
- gptr
  - \_\_gnu\_cxx::enc\_filebuf, 718
  - \_\_gnu\_cxx::stdio\_filebuf, 769
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 788
  - std::basic\_filebuf, 1554
  - std::basic\_streambuf, 1918
  - std::basic\_stringbuf, 1979
- grep
  - std::regex\_constants, 615
- grouping
  - std::moneypunct, 2492
  - std::moneypunct\_byname, 2501
  - std::numpunct, 2599
  - std::numpunct\_byname, 2604
- gslice
  - Numeric Arrays, 83, 84
- gslice.h, 3048

- gslice\_array
  - Numeric Arrays, [84](#)
- gslice\_array.h, [3048](#)
- has\_denorm
  - std::\_\_numeric\_limits\_base, [1421](#)
  - std::numeric\_limits, [2574](#)
- has\_denorm\_loss
  - std::\_\_numeric\_limits\_base, [1421](#)
  - std::numeric\_limits, [2574](#)
- has\_facet
  - Locales, [179](#)
  - std::locale, [2414](#)
  - std::locale::id, [2419](#)
- has\_infinity
  - std::\_\_numeric\_limits\_base, [1422](#)
  - std::numeric\_limits, [2574](#)
- has\_quiet\_NaN
  - std::\_\_numeric\_limits\_base, [1422](#)
  - std::numeric\_limits, [2574](#)
- has\_signaling\_NaN
  - std::\_\_numeric\_limits\_base, [1422](#)
  - std::numeric\_limits, [2574](#)
- hash
  - std::collate, [2092](#)
  - std::collate\_byname, [2096](#)
- hash<\_Tp>, [1270](#)
- Hash-Based, [249](#)
- hash\_bytes.h, [3049](#)
- hash\_eq\_fn<Key, Eq\_Fn, \_Alloc, Store\_Hash>, [1271](#)
- hash\_eq\_fn.hpp, [3049](#)
- hash\_exponential\_size\_policy
  - \_\_gnu\_pbds::hash\_exponential\_size\_policy, [1180](#)
- hash\_exponential\_size\_policy\_imp.hpp, [3050](#)
- hash\_fun.h, [3050](#)
- hash\_function
  - std::unordered\_map, [2798](#)
  - std::unordered\_multimap, [2817](#)
  - std::unordered\_multiset, [2835](#)
  - std::unordered\_set, [2853](#)
- hash\_load\_check\_resize\_trigger
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, [1181](#)
- hash\_load\_check\_resize\_trigger\_imp.hpp, [3050](#)
- hash\_load\_check\_resize\_trigger\_size\_base<Size\_Type, Hold\_Size>, [1271](#)
- hash\_load\_check\_resize\_trigger\_size\_base.hpp, [3050](#)
- hash\_map, [3051](#)
- hash\_policy.hpp, [3052](#)
- hash\_prime\_size\_policy
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, [1183](#)
- hash\_prime\_size\_policy\_imp.hpp, [3053](#)
- hash\_set, [3053](#)
- hash\_standard\_resize\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy, [1184](#)
- hash\_standard\_resize\_policy\_imp.hpp, [3054](#)
- hasher
  - std::unordered\_map, [2787](#)
  - std::unordered\_multimap, [2807](#)
  - std::unordered\_multiset, [2825](#)
  - std::unordered\_set, [2843](#)
- Hashes, [173](#)
- hashtable.h, [3054](#), [3055](#)
- hashtable\_policy.h, [3055](#)
- Heap, [225](#)
  - is\_heap, [225](#), [226](#)
  - is\_heap\_until, [226](#)
  - make\_heap, [227](#)
  - pop\_heap, [227](#), [228](#)
  - push\_heap, [228](#)
  - sort\_heap, [229](#)
- Heap-Based, [255](#)
  - priority\_queue, [256](#)
- hermite
  - Mathematical Special Functions, [245](#)
- hex
  - std, [518](#)
  - std::basic\_fstream, [1615](#)
  - std::basic\_ifstream, [1655](#)
  - std::basic\_ios, [1679](#)
  - std::basic\_iostream, [1725](#)
  - std::basic\_istream, [1764](#)
  - std::basic\_istreamstringstream, [1804](#)
  - std::basic\_ofstream, [1837](#)
  - std::basic\_ostream, [1868](#)
  - std::basic\_ostreamstringstream, [1902](#)
  - std::basic\_stringstream, [2037](#)
  - std::ios\_base, [2316](#)
- hours
  - std::chrono, [599](#)
- hyperg
  - Mathematical Special Functions, [245](#)
- I/O, [42](#)
  - filebuf, [43](#)
  - fstream, [43](#)
  - ifstream, [43](#)
  - ios, [43](#)
  - iostream, [43](#)
  - istream, [43](#)
  - istreamstringstream, [44](#)
  - ofstream, [44](#)
  - ostream, [44](#)
  - ostreamstringstream, [44](#)
  - streambuf, [44](#)
  - stringbuf, [44](#)
  - stringstream, [44](#)
  - wfilebuf, [44](#)
  - wfstream, [44](#)

- wifstream, [44](#)
- wios, [45](#)
- wiostream, [45](#)
- wistream, [45](#)
- wistreamstream, [45](#)
- wofstream, [45](#)
- wostream, [45](#)
- wostringstream, [45](#)
- wstreambuf, [45](#)
- wstringbuf, [45](#)
- wstringstream, [45](#)
- icase
  - std::regex\_constants, [615](#)
- id
  - std::collate, [2092](#)
  - std::collate\_byname, [2097](#)
  - std::ctype, [2119](#)
  - std::ctype< char >, [2130](#)
  - std::ctype< wchar\_t >, [2143](#)
  - std::ctype\_byname, [2155](#)
  - std::ctype\_byname< char >, [2166](#)
  - std::locale::id, [2419](#)
  - std::messages, [2470](#)
  - std::messages\_byname, [2473](#)
  - std::money\_get, [2481](#)
  - std::money\_put, [2486](#)
  - std::moneypunct, [2495](#)
  - std::moneypunct\_byname, [2503](#)
  - std::num\_get, [2560](#)
  - std::num\_put, [2570](#)
  - std::numpunct, [2600](#)
  - std::numpunct\_byname, [2605](#)
  - std::time\_get, [2727](#)
  - std::time\_get\_byname, [2735](#)
  - std::time\_put, [2739](#)
  - std::time\_put\_byname, [2742](#)
- identity\_element
  - SGL, [13](#)
- ifstream
  - I/O, [43](#)
- ignore
  - std::basic\_fstream, [1588](#)
  - std::basic\_ifstream, [1634](#), [1635](#)
  - std::basic\_iostream, [1698](#), [1699](#)
  - std::basic\_istream, [1743](#), [1744](#)
  - std::basic\_istreamstream, [1784](#)
  - std::basic\_stringstream, [2010](#), [2011](#)
- imbue
  - \_\_gnu\_cxx::enc\_filebuf, [718](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [770](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [789](#)
  - std::basic\_filebuf, [1555](#)
  - std::basic\_fstream, [1589](#)
  - std::basic\_ifstream, [1636](#)
  - std::basic\_ios, [1669](#)
  - std::basic\_iostream, [1699](#)
  - std::basic\_istream, [1744](#)
  - std::basic\_istreamstream, [1785](#)
  - std::basic\_ofstream, [1819](#)
  - std::basic\_ostream, [1851](#)
  - std::basic\_ostreamstream, [1884](#)
  - std::basic\_regex, [1910](#)
  - std::basic\_streambuf, [1918](#)
  - std::basic\_stringbuf, [1979](#)
  - std::basic\_stringstream, [2011](#)
  - std::ios\_base, [2310](#)
  - std::regex\_traits, [2662](#)
- in
  - std::\_\_codecvt\_abstract\_base, [1289](#)
  - std::basic\_fstream, [1616](#)
  - std::basic\_ifstream, [1656](#)
  - std::basic\_ios, [1679](#)
  - std::basic\_iostream, [1725](#)
  - std::basic\_istream, [1764](#)
  - std::basic\_istreamstream, [1805](#)
  - std::basic\_ofstream, [1838](#)
  - std::basic\_ostream, [1869](#)
  - std::basic\_ostreamstream, [1902](#)
  - std::basic\_stringstream, [2038](#)
  - std::codecvt, [2070](#)
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, [2073](#)
  - std::codecvt< char, char, mbstate\_t >, [2077](#)
  - std::codecvt< wchar\_t, char, mbstate\_t >, [2081](#)
  - std::codecvt\_byname, [2086](#)
  - std::ios\_base, [2316](#)
- in\_avail
  - \_\_gnu\_cxx::enc\_filebuf, [719](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [770](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [789](#)
  - std::basic\_filebuf, [1555](#)
  - std::basic\_streambuf, [1919](#)
  - std::basic\_stringbuf, [1980](#)
- includes
  - Set Operation, [151](#)
- increment
  - std::linear\_congruential\_engine, [2390](#)
- independent\_bits\_engine
  - std::independent\_bits\_engine, [2292](#), [2293](#)
- indirect\_array
  - Numeric Arrays, [84](#)
- indirect\_array.h, [3058](#)
- infinity
  - std::numeric\_limits, [2572](#)
- info\_fn\_imps.hpp, [3058d](#)
- init
  - std::basic\_fstream, [1589](#)
  - std::basic\_ifstream, [1636](#)

- std::basic\_ios, 1669
- std::basic\_iostream, 1699
- std::basic\_istream, 1745
- std::basic\_istream, 1785
- std::basic\_ofstream, 1820
- std::basic\_ostream, 1852
- std::basic\_ostream, 1885
- std::basic\_stringstream, 2011
- initializer\_list, 3060
- inner\_product
  - std, 518
- inplace\_merge
  - Sorting, 135
- insert
  - \_\_gnu\_cxx::\_\_versa\_string, 676d
  - \_\_gnu\_debug::basic\_string, 865d
  - std::\_\_detail::\_\_Nfa, 1400, 1401
  - std::basic\_string, 1957d
  - std::deque, 2188, 2189
  - std::list, 2401, 2402
  - std::map, 2441, 2442
  - std::multimap, 2514, 2515
  - std::multiset, 2529, 2530
  - std::set, 2688, 2689
  - std::unordered\_map, 2798d
  - std::unordered\_multimap, 2817d
  - std::unordered\_multiset, 2835d
  - std::unordered\_set, 2853d
  - std::vector, 2875, 2876
- insert\_after
  - std::forward\_list, 2233, 2234
- insert\_fn\_imps.hpp, 3060d
- insert\_iterator
  - std::insert\_iterator, 2300
- insert\_join\_fn\_imps.hpp, 3063
- insert\_no\_store\_hash\_fn\_imps.hpp, 3063
- insert\_store\_hash\_fn\_imps.hpp, 3063
- inserter
  - Iterators, 234
- int\_type
  - std::basic\_ios, 1663
  - std::basic\_streambuf, 1915
  - std::istreambuf\_iterator, 2377
- internal
  - std, 518
  - std::basic\_fstream, 1616
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1679
  - std::basic\_iostream, 1726
  - std::basic\_istream, 1764
  - std::basic\_istream, 1805
  - std::basic\_ofstream, 1838
  - std::basic\_ostream, 1869
  - std::basic\_ostream, 1902
  - std::basic\_stringstream, 2038
  - std::ios\_base, 2317
- intervals
  - std::piecewise\_constant\_distribution, 2620
  - std::piecewise\_linear\_distribution, 2625
- intl
  - std::moneypunct, 2495
- Invalidation Guarantees, 260
- iomanip, 3063
- ios, 3065
  - I/O, 43
- ios\_base.h, 3065
- iosfwd, 3067
- iostate
  - std::basic\_fstream, 1577
  - std::basic\_ifstream, 1624
  - std::basic\_ios, 1663
  - std::basic\_iostream, 1688
  - std::basic\_istream, 1733
  - std::basic\_istream, 1774
  - std::basic\_ofstream, 1812
  - std::basic\_ostream, 1845
  - std::basic\_ostream, 1878
  - std::basic\_stringstream, 1999
  - std::ios\_base, 2307
- iostream, 3068
  - I/O, 43
- iota
  - std, 519
- is
  - std::\_\_ctype\_abstract\_base, 1298
  - std::ctype, 2115
  - std::ctype< char >, 2126
  - std::ctype< wchar\_t >, 2138, 2139
  - std::ctype\_byname, 2151
  - std::ctype\_byname< char >, 2161
- is\_bounded
  - std::\_\_numeric\_limits\_base, 1422
  - std::numeric\_limits, 2574
- is\_emptyset
  - std::tr2::bool\_set, 2746
- is\_exact
  - std::\_\_numeric\_limits\_base, 1422
  - std::numeric\_limits, 2574
- is\_grow\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, 995
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1212
- is\_heap
  - Heap, 225, 226
- is\_heap\_until
  - Heap, 226
- is\_iec559
  - std::\_\_numeric\_limits\_base, 1422

- std::numeric\_limits, 2574
- is\_indeterminate
  - std::tr2::bool\_set, 2746
- is\_integer
  - std::\_\_numeric\_limits\_base, 1422
  - std::numeric\_limits, 2574
- is\_modulo
  - std::\_\_numeric\_limits\_base, 1422
  - std::numeric\_limits, 2574
- is\_open
  - \_\_gnu\_cxx::enc\_filebuf, 719
  - \_\_gnu\_cxx::stdio\_filebuf, 770
  - std::basic\_filebuf, 1555
  - std::basic\_fstream, 1589
  - std::basic\_ifstream, 1636
  - std::basic\_ofstream, 1820
- is\_partitioned
  - Mutating, 109
- is\_permutation
  - Non-Mutating, 128
- is\_resize\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, 995
  - \_\_gnu\_pbds::sample\_resize\_policy, 1210
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1212
- is\_signed
  - std::\_\_numeric\_limits\_base, 1422
  - std::numeric\_limits, 2575
- is\_singleton
  - std::tr2::bool\_set, 2746
- is\_sorted
  - Sorting, 136
- is\_sorted\_until
  - Sorting, 136, 137
- is\_specialized
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2575
- isalnum
  - std, 519
- isalpha
  - std, 519
- iscntrl
  - std, 519
- isctype
  - Regular Expressions, 188
- isdigit
  - std, 519
- isgraph
  - std, 519
- islower
  - std, 519
- isprint
  - std, 519
- ispunct
  - std, 520
- isspace
  - std, 520
- istream, 3068
  - I/O, 43
- istream.tcc, 3069
- istream\_iterator
  - std::istream\_iterator, 2375
- istream\_type
  - std::istreambuf\_iterator, 2377
- istreambuf\_iterator
  - std::istreambuf\_iterator, 2378
- istringstream
  - I/O, 44
- isupper
  - std, 520
- isxdigit
  - std, 520
- iter\_swap
  - Mutating, 109
- iter\_type
  - std::money\_get, 2479
  - std::money\_put, 2483
  - std::num\_get, 2548
  - std::num\_put, 2562
  - std::time\_get, 2720
  - std::time\_put, 2736
- iterator, 3070
  - std::set, 2680
  - std::unordered\_map, 2787
  - std::unordered\_multimap, 2807
  - std::unordered\_multiset, 2825
  - std::unordered\_set, 2843
- Iterator Tags, 236
- iterator.h, 3071
- iterator.hpp, 3071
- iterator\_, 1272
  - const\_pointer, 1273
  - const\_reference, 1273
  - difference\_type, 1273
  - iterator\_, 1274
  - iterator\_category, 1273
  - iterator\_, 1274
  - m\_p\_tbl, 1275
  - operator const point\_iterator\_, 1274
  - operator point\_iterator\_, 1274
  - operator\*, 1274
  - operator++, 1274, 1275
  - operator->, 1275
  - operator==, 1275
  - pointer, 1273
  - reference, 1273
  - value\_type, 1273
- iterator\_category

- \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator, [1014](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator, [1019](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator, [1030](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator, [1033](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator, [1074](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator, [1078](#)
  - const\_iterator, [1266](#)
  - iterator, [1273](#)
  - point\_const\_iterator, [1277](#)
  - point\_iterator, [1280](#)
  - std::back\_insert\_iterator, [1541](#)
  - std::front\_insert\_iterator, [2243](#)
  - std::insert\_iterator, [2300](#)
  - std::istream\_iterator, [2374](#)
  - std::istreambuf\_iterator, [2377](#)
  - std::iterator, [2381](#)
  - std::ostream\_iterator, [2607](#)
  - std::ostreambuf\_iterator, [2610](#)
  - std::raw\_storage\_iterator, [2651](#)
  - std::reverse\_iterator, [2670](#)
- iterator\_fn\_imps.hpp, [3071](#)
- iterator\_tracker.h, [3072](#)
- Iterators, [230](#)
  - \_\_iterator\_category, [233](#)
  - back\_inserter, [233](#)
  - front\_inserter, [233](#)
  - inserter, [234](#)
  - operator==, [234](#)
- iterators\_fn\_imps.hpp, [3073](#), [3074](#)
- iword
  - std::basic\_fstream, [1590](#)
  - std::basic\_ifstream, [1636](#)
  - std::basic\_ios, [1669](#)
  - std::basic\_iostream, [1700](#)
  - std::basic\_istream, [1745](#)
  - std::basic\_istreamstream, [1785](#)
  - std::basic\_ofstream, [1820](#)
  - std::basic\_ostream, [1852](#)
  - std::basic\_ostreamstream, [1885](#)
  - std::basic\_stringstream, [2012](#)
  - std::ios\_base, [2310](#)
- k
  - std::negative\_binomial\_distribution, [2537](#)
- key\_comp
  - std::map, [2442](#)
  - std::multimap, [2515](#)
  - std::multiset, [2531](#)
  - std::set, [2689](#)
- key\_compare
  - std::set, [2680](#)
- key\_eq
  - std::unordered\_map, [2800](#)
  - std::unordered\_multimap, [2820](#)
  - std::unordered\_multiset, [2837](#)
  - std::unordered\_set, [2856](#)
- key\_equal
  - std::unordered\_map, [2788](#)
  - std::unordered\_multimap, [2807](#)
  - std::unordered\_multiset, [2826](#)
  - std::unordered\_set, [2843](#)
- key\_type
  - std::set, [2680](#)
  - std::unordered\_map, [2788](#)
  - std::unordered\_multimap, [2807](#)
  - std::unordered\_multiset, [2826](#)
  - std::unordered\_set, [2843](#)
- kill\_dependency
  - Atomics, [172](#)
- L1\_cache\_size
  - \_\_gnu\_parallel::\_Settings, [962](#)
- L2\_cache\_size
  - \_\_gnu\_parallel::\_Settings, [962](#)
- laguerre
  - Mathematical Special Functions, [245](#)
- lambda
  - std::exponential\_distribution, [2212](#)
- launch
  - Futures, [41](#)
- left
  - std, [520](#)
  - std::basic\_fstream, [1616](#)
  - std::basic\_ifstream, [1656](#)
  - std::basic\_ios, [1679](#)
  - std::basic\_iostream, [1726](#)
  - std::basic\_istream, [1764](#)
  - std::basic\_istreamstream, [1805](#)
  - std::basic\_ofstream, [1838](#)
  - std::basic\_ostream, [1869](#)
  - std::basic\_ostreamstream, [1902](#)
  - std::basic\_stringstream, [2038](#)
  - std::ios\_base, [2317](#)
- left\_child\_next\_sibling\_heap\_.hpp, [3074](#)
- legendre
  - Mathematical Special Functions, [245](#)
- length
  - \_\_gnu\_cxx::\_versa\_string, [680](#)
  - \_\_gnu\_debug::basic\_string, [867](#)
  - std::basic\_string, [1961](#)
  - std::match\_results, [2455](#)
  - std::regex\_traits, [2662](#)
  - std::sub\_match, [2714](#)

- lexicographical\_compare
  - Sorting, 137
- lexicographical\_compare\_3way
  - SGL, 13
- limits, 3074
- linear\_congruential\_engine
  - std::linear\_congruential\_engine, 2387, 2388
- linear\_probe\_fn\_imp.hpp, 3076
- list, 3076, 3077
  - std::list, 2394d
- List-Based, 253
- list.tcc, 3078
- list\_partition
  - \_\_gnu\_parallel, 358
- list\_partition.h, 3078
- list\_update
  - \_\_gnu\_pbds::list\_update, 1189
- list\_update\_policy.hpp, 3079
- load\_factor
  - std::unordered\_map, 2801
  - std::unordered\_multimap, 2820
  - std::unordered\_multiset, 2837
  - std::unordered\_set, 2856
- local\_iterator
  - std::unordered\_map, 2788
  - std::unordered\_multimap, 2808
  - std::unordered\_multiset, 2826
  - std::unordered\_set, 2844
- locale, 3079
  - std::locale, 2410, 2411
- locale\_classes.h, 3079
- locale\_classes.tcc, 3080
- locale\_facets.h, 3081
- locale\_facets.tcc, 3082
- locale\_facets\_nonio.h, 3083
- locale\_facets\_nonio.tcc, 3084
- localefwd.h, 3084
- Locales, 178
  - has\_facet, 179
  - use\_facet, 180
- lock
  - Mutexes, 53
- log
  - Complex Numbers, 33
- log10
  - Complex Numbers, 33
- logic\_error
  - std::logic\_error, 2421
- lookup\_classname
  - std::regex\_traits, 2662
- lookup\_collatename
  - std::regex\_traits, 2663
- losertree.h, 3087
- lower\_bound
  - Binary Search, 158, 159
  - std::map, 2442, 2443
  - std::multimap, 2515, 2516
  - std::multiset, 2531
  - std::set, 2689
- lowest
  - std::numeric\_limits, 2572
- lu\_counter\_metadata.hpp, 3088
- lu\_map.hpp, 3088
- m\_p\_tbl
  - const\_iterator\_, 1268
  - iterator\_, 1275
- macros.h, 3089
  - \_\_glibcxx\_check\_erase, 3089
  - \_\_glibcxx\_check\_erase\_after, 3089
  - \_\_glibcxx\_check\_erase\_range, 3090
  - \_\_glibcxx\_check\_erase\_range\_after, 3090
  - \_\_glibcxx\_check\_heap\_pred, 3090
  - \_\_glibcxx\_check\_insert, 3090
  - \_\_glibcxx\_check\_insert\_after, 3090
  - \_\_glibcxx\_check\_insert\_range, 3090
  - \_\_glibcxx\_check\_insert\_range\_after, 3090
  - \_\_glibcxx\_check\_partitioned\_lower, 3091
  - \_\_glibcxx\_check\_partitioned\_lower\_pred, 3091
  - \_\_glibcxx\_check\_partitioned\_upper\_pred, 3091
  - \_\_glibcxx\_check\_sorted\_pred, 3091
- make\_error\_code
  - Futures, 41
- make\_error\_condition
  - Futures, 41
- make\_exception\_ptr
  - Exceptions, 25
- make\_heap
  - Heap, 227
- make\_pair
  - Utilities, 68
- make\_shared
  - Pointer Abstractions, 51
- malloc\_allocator.h, 3091
- map, 3092
  - std::map, 2433, 2434
- map.h, 3093
- mapped\_type
  - std::unordered\_map, 2788
  - std::unordered\_multimap, 2808
- mark\_count
  - std::basic\_regex, 1910
- mask\_array
  - Numeric Arrays, 84
- mask\_array.h, 3094
- mask\_based\_range\_hashing.hpp, 3095
- match\_any
  - std::regex\_constants, 615

- match\_continuous
  - std::regex\_constants, 615
- match\_default
  - std::regex\_constants, 615
- match\_flag\_type
  - std::regex\_constants, 612
- match\_not\_bol
  - std::regex\_constants, 616
- match\_not\_bow
  - std::regex\_constants, 616
- match\_not\_eol
  - std::regex\_constants, 616
- match\_not\_eow
  - std::regex\_constants, 616
- match\_not\_null
  - std::regex\_constants, 616
- match\_prev\_avail
  - std::regex\_constants, 616
- match\_results
  - std::match\_results, 2452, 2453
- Mathematical Special Functions, 241
  - assoc\_laguerre, 243
  - assoc\_legendre, 243
  - beta, 243
  - comp\_ellint\_1, 243
  - comp\_ellint\_2, 243
  - comp\_ellint\_3, 243
  - conf\_hyperg, 244
  - cyl\_bessel\_i, 244
  - cyl\_bessel\_j, 244
  - cyl\_bessel\_k, 244
  - cyl\_neumann, 244
  - ellint\_1, 244
  - ellint\_2, 244
  - ellint\_3, 244
  - expint, 245
  - hermite, 245
  - hyperg, 245
  - laguerre, 245
  - legendre, 245
  - riemann\_zeta, 245
  - sph\_bessel, 245
  - sph\_legendre, 245
  - sph\_neumann, 245
- max
  - \_\_gnu\_parallel, 359
  - Numeric Arrays, 87
  - Sorting, 138
  - std::bernoulli\_distribution, 2041
  - std::binomial\_distribution, 2051
  - std::cauchy\_distribution, 2055
  - std::chi\_squared\_distribution, 2062
  - std::discard\_block\_engine, 2196
  - std::discrete\_distribution, 2200
  - std::exponential\_distribution, 2212
  - std::extreme\_value\_distribution, 2217
  - std::fisher\_f\_distribution, 2220
  - std::gamma\_distribution, 2254
  - std::geometric\_distribution, 2257
  - std::independent\_bits\_engine, 2294
  - std::linear\_congruential\_engine, 2388
  - std::lognormal\_distribution, 2427
  - std::mersenne\_twister\_engine, 2466
  - std::negative\_binomial\_distribution, 2537
  - std::normal\_distribution, 2542
  - std::numeric\_limits, 2572
  - std::piecewise\_constant\_distribution, 2621
  - std::piecewise\_linear\_distribution, 2625
  - std::poisson\_distribution, 2635
  - std::shuffle\_order\_engine, 2701
  - std::student\_t\_distribution, 2709
  - std::uniform\_int\_distribution, 2775
  - std::uniform\_real\_distribution, 2779
  - std::weibull\_distribution, 2886
- max\_count
  - \_\_gnu\_pbds::lu\_counter\_policy, 1192
- max\_bucket\_count
  - std::unordered\_map, 2801
  - std::unordered\_multimap, 2820
  - std::unordered\_multiset, 2838
  - std::unordered\_set, 2856
- max\_digits10
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2575
- max\_element
  - Sorting, 139
- max\_element\_minimal\_n
  - \_\_gnu\_parallel::Settings, 962
- max\_exponent
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2575
- max\_exponent10
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2575
- max\_load\_factor
  - std::unordered\_map, 2801
  - std::unordered\_multimap, 2820
  - std::unordered\_multiset, 2838
  - std::unordered\_set, 2856
- max\_size
  - \_\_gnu\_cxx::\_\_alloc\_traits, 632
  - \_\_gnu\_cxx::\_\_versa\_string, 680
  - \_\_gnu\_debug::basic\_string, 868
  - std::\_\_detail::\_\_Nfa, 1401
  - std::allocator\_traits, 1499
  - std::basic\_string, 1962
  - std::deque, 2189
  - std::forward\_list, 2235

- std::list, [2402](#)
- std::map, [2443](#)
- std::match\_results, [2455](#)
- std::multimap, [2516](#)
- std::multiset, [2531](#)
- std::set, [2690](#)
- std::tr2::dynamic\_bitset, [2755](#)
- std::unordered\_map, [2801](#)
- std::unordered\_multimap, [2820](#)
- std::unordered\_multiset, [2838](#)
- std::unordered\_set, [2856](#)
- std::vector, [2877](#)
- mean
  - std::normal\_distribution, [2542](#)
  - std::poisson\_distribution, [2635](#)
- mem\_fn
  - Function Objects, [215](#)
- Memory, [47](#)
- memory, [3095](#)
- memory\_order
  - Atomics, [172](#)
- merge
  - Sorting, [139](#), [140](#)
  - std::forward\_list, [2235](#)
  - std::list, [2402](#)
- merge.h, [3096](#)
- merge\_minimal\_n
  - \_\_gnu\_parallel:: Settings, [962](#)
- merge\_oversampling
  - \_\_gnu\_parallel:: Settings, [963](#)
- mersenne\_twister\_engine
  - std::mersenne\_twister\_engine, [2465](#)
- messages
  - std::locale, [2415](#)
  - std::messages, [2470](#)
- messages\_members.h, [3097](#)
- metadata\_const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, [1014](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, [1019](#)
- metadata\_reference
  - \_\_gnu\_pbds::lu\_counter\_policy, [1192](#)
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy, [1193](#)
- metadata\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, [1014](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, [1020](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, [1112](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, [1116](#)
  - \_\_gnu\_pbds::lu\_counter\_policy, [1192](#)
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy, [1193](#)
  - \_\_gnu\_pbds::sample\_update\_policy, [1218](#)
- Metaprogramming, [59](#)
- false\_type, [63](#)
- true\_type, [63](#)
- microseconds
  - std::chrono, [599](#)
- milliseconds
  - std::chrono, [599](#)
- min
  - \_\_gnu\_parallel, [359](#)
  - Numeric Arrays, [87](#)
  - Sorting, [140](#), [141](#)
  - std::bernoulli\_distribution, [2041](#)
  - std::binomial\_distribution, [2051](#)
  - std::cauchy\_distribution, [2055](#)
  - std::chi\_squared\_distribution, [2062](#)
  - std::discard\_block\_engine, [2196](#)
  - std::discrete\_distribution, [2200](#)
  - std::exponential\_distribution, [2212](#)
  - std::extreme\_value\_distribution, [2217](#)
  - std::fisher\_f\_distribution, [2220](#)
  - std::gamma\_distribution, [2254](#)
  - std::geometric\_distribution, [2257](#)
  - std::independent\_bits\_engine, [2294](#)
  - std::linear\_congruential\_engine, [2388](#)
  - std::lognormal\_distribution, [2427](#)
  - std::mersenne\_twister\_engine, [2466](#)
  - std::negative\_binomial\_distribution, [2537](#)
  - std::normal\_distribution, [2542](#)
  - std::numeric\_limits, [2573](#)
  - std::piecewise\_constant\_distribution, [2621](#)
  - std::piecewise\_linear\_distribution, [2625](#)
  - std::poisson\_distribution, [2635](#)
  - std::shuffle\_order\_engine, [2701](#)
  - std::student\_t\_distribution, [2709](#)
  - std::uniform\_int\_distribution, [2775](#)
  - std::uniform\_real\_distribution, [2779](#)
  - std::weibull\_distribution, [2887](#)
- min\_element
  - Sorting, [141](#)
- min\_element\_minimal\_n
  - \_\_gnu\_parallel:: Settings, [963](#)
- min\_exponent
  - std::\_\_numeric\_limits\_base, [1423](#)
  - std::numeric\_limits, [2575](#)
- min\_exponent10
  - std::\_\_numeric\_limits\_base, [1423](#)
  - std::numeric\_limits, [2575](#)
- minmax
  - Sorting, [142](#)
- minmax\_element
  - Sorting, [142](#)
- minstd\_rand
  - Random Number Generators, [267](#)
- minstd\_rand0
  - Random Number Generators, [268](#)

- minutes
  - std::chrono, 600
- mismatch
  - Non-Mutating, 129
- mod\_based\_range\_hashing.hpp, 3097
- modulus
  - std::linear\_congruential\_engine, 2390
- monetary
  - std::locale, 2415
- money\_get
  - std::money\_get, 2479
- money\_put
  - std::money\_put, 2483
- money\_punct
  - std::money\_punct, 2488, 2489
- move
  - Mutating, 109
  - Utilities, 68
- move.h, 3097
- move\_backward
  - Mutating, 110
- move\_if\_noexcept
  - Utilities, 69
- mt19937
  - Random Number Generators, 268
- mt19937\_64
  - Random Number Generators, 268
- mt\_allocator.h, 3098
- multimap
  - std::multimap, 2507, 2508
- multimap.h, 3099, 3100
- multiplier
  - std::linear\_congruential\_engine, 2390
- multisec\_partition
  - \_\_gnu\_parallel, 359
- multisec\_selection
  - \_\_gnu\_parallel, 360
- multisec\_selection.h, 3101
- multiset
  - std::multiset, 2523, 2524
- multiset.h, 3102, 3103
- multiway\_merge
  - \_\_gnu\_parallel, 360
- multiway\_merge.h, 3103
- multiway\_merge\_3\_variant
  - \_\_gnu\_parallel, 361
- multiway\_merge\_4\_variant
  - \_\_gnu\_parallel, 362
- multiway\_merge\_exact\_splitting
  - \_\_gnu\_parallel, 363
- multiway\_merge\_loser\_tree
  - \_\_gnu\_parallel, 363
- multiway\_merge\_loser\_tree\_sentinel
  - \_\_gnu\_parallel, 363
- multiway\_merge\_loser\_tree\_unguarded
  - \_\_gnu\_parallel, 364
- multiway\_merge\_minimal\_k
  - \_\_gnu\_parallel::Settings, 963
- multiway\_merge\_minimal\_n
  - \_\_gnu\_parallel::Settings, 963
- multiway\_merge\_oversampling
  - \_\_gnu\_parallel::Settings, 963
- multiway\_merge\_sampling\_splitting
  - \_\_gnu\_parallel, 364
- multiway\_merge\_sentinels
  - \_\_gnu\_parallel, 365
- multiway\_mergesort.h, 3107
- Mutating, 104
  - copy, 106
  - copy\_backward, 106
  - copy\_if, 106
  - copy\_n, 107
  - fill, 107
  - fill\_n, 108
  - generate, 108
  - generate\_n, 108
  - is\_partitioned, 109
  - iter\_swap, 109
  - move, 109
  - move\_backward, 110
  - partition, 110
  - partition\_copy, 111
  - partition\_point, 111
  - random\_shuffle, 112
  - remove, 112
  - remove\_copy, 113
  - remove\_copy\_if, 113
  - remove\_if, 114
  - replace, 114
  - replace\_copy\_if, 114
  - replace\_if, 115
  - reverse, 115
  - reverse\_copy, 115
  - rotate, 116
  - rotate\_copy, 116
  - shuffle, 117
  - stable\_partition, 117
  - swap\_ranges, 118
  - transform, 118
  - unique, 119
  - unique\_copy, 120
- mutex, 3107
- Mutexes, 52
  - call\_once, 53
  - lock, 53
  - swap, 53
  - try\_lock, 53

- name
  - std::locale, [2412](#)
  - std::type\_info, [2769](#)
- nanoseconds
  - std::chrono, [600](#)
- narrow
  - std::\_\_ctype\_abstract\_base, [1298](#), [1299](#)
  - std::basic\_fstream, [1590](#)
  - std::basic\_ifstream, [1636](#)
  - std::basic\_ios, [1670](#)
  - std::basic\_iostream, [1700](#)
  - std::basic\_istream, [1745](#)
  - std::basic\_istreamstream, [1786](#)
  - std::basic\_ofstream, [1821](#)
  - std::basic\_ostream, [1852](#)
  - std::basic\_ostreamstream, [1885](#)
  - std::basic\_stringstream, [2012](#)
  - std::ctype, [2116](#)
  - std::ctype< char >, [2126](#), [2127](#)
  - std::ctype< wchar\_t >, [2139](#)
  - std::ctype\_byname, [2151](#), [2152](#)
  - std::ctype\_byname< char >, [2162](#)
- native\_handle
  - std::thread, [2717](#)
- neg\_format
  - std::moneypunct, [2493](#)
  - std::moneypunct\_byname, [2501](#)
- negative\_sign
  - std::moneypunct, [2493](#)
  - std::moneypunct\_byname, [2501](#)
- Negators, [219](#)
  - not1, [219](#)
  - not2, [220](#)
- nested\_exception.h, [3109](#)
- new, [3109](#)
  - operator delete, [3110](#), [3111](#)
  - operator new, [3112](#)
- new\_allocator.h, [3113](#)
- new\_handler
  - std, [496](#)
- next\_permutation
  - Sorting, [143](#)
- noboolalpha
  - std, [520](#)
- node.hpp, [3114](#), [3115](#)
- node\_begin
  - \_\_gnu\_pbds::detail::ov\_tree\_map, [1088](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map, [1120](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map, [1129](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map, [1139](#)
- node\_const\_iterator
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits, [1024](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [1025](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [1149](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [1152](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [1155](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [1156](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [1158](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [1161](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_A-Traits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1165](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_A-Traits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1167](#)
- node\_end
  - \_\_gnu\_pbds::detail::ov\_tree\_map, [1089](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map, [1120](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map, [1130](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map, [1139](#)
- node\_iterators.hpp, [3115](#)
- node\_metadata\_selector.hpp, [3116](#)
- node\_type
  - \_\_gnu\_pbds::detail::pat\_trie\_base, [1098](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map, [1120](#)
- node\_update
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_A-Traits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1166](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_A-Traits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1167](#)
- Non-Mutating, [121](#)
  - adjacent\_find, [122](#)
  - all\_of, [123](#)
  - any\_of, [123](#)
  - count, [124](#)
  - count\_if, [124](#)
  - equal, [124](#), [125](#)
  - find, [125](#)
  - find\_end, [125](#), [126](#)
  - find\_first\_of, [126](#), [127](#)
  - find\_if, [127](#)
  - find\_if\_not, [127](#)
  - for\_each, [128](#)
  - is\_permutation, [128](#)

- mismatch, [129](#)
- none\_of, [130](#)
- search, [130](#)
- search\_n, [131](#)
- none
  - std, [520](#)
  - std::locale, [2416](#)
  - std::tr2::dynamic\_bitset, [2755](#)
- none\_of
  - Non-Mutating, [130](#)
- norm
  - Complex Numbers, [33](#)
- Normal Distributions, [275](#)
  - operator<<, [277](#)
  - operator>>, [277](#)
- normal\_distribution
  - std::normal\_distribution, [2542](#)
- noshowbase
  - std, [520](#)
- noshowpoint
  - std, [521](#)
- noshowpos
  - std, [521](#)
- noskipws
  - std, [521](#)
- nosubs
  - std::regex\_constants, [616](#)
- not1
  - Negators, [219](#)
- not2
  - Negators, [220](#)
- notify\_cleared
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [995](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, [1182](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1210](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_erase\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [995](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1210](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_erase\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [995](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1210](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_erase\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [995](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1210](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_erased
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [995](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1210](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_externally\_resized
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_find\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_find\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_find\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_insert\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_insert\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_insert\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [996](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_inserted
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [997](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, [1182](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- notify\_resized
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, [997](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, [1182](#)
  - \_\_gnu\_pbds::sample\_range\_hashing, [1207](#)
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, [1208](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1211](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1213](#)
- nounitbuf
  - std, [521](#)

- nouppercase
  - std, [521](#)
- npos
  - \_\_gnu\_cxx::\_\_versa\_string, [695](#)
  - \_\_gnu\_debug::basic\_string, [873](#)
  - std::basic\_string, [1974](#)
- nth\_element
  - Sorting, [144](#)
- nth\_element\_minimal\_n
  - \_\_gnu\_parallel::Settings, [963](#)
- null\_node\_metadata.hpp, [3117](#)
- num\_blocks
  - std::tr2::dynamic\_bitset, [2755](#)
- num\_children
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, [1113](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, [1116](#)
- num\_get
  - std::num\_get, [2548](#)
- num\_put
  - std::num\_put, [2562](#)
- numeric, [3117](#), [3118](#)
  - std::locale, [2416](#)
- Numeric Arrays, [72](#)
  - ~gslice, [85](#)
  - apply, [85](#), [86](#)
  - begin, [86](#)
  - cshift, [87](#)
  - end, [87](#)
  - gslice, [83](#), [84](#)
  - gslice\_array, [84](#)
  - indirect\_array, [84](#)
  - mask\_array, [84](#)
  - max, [87](#)
  - min, [87](#)
  - operator<=, [91](#), [92](#)
  - operator>=, [95](#), [96](#)
  - operator\*=, [89](#)
  - operator~, [100](#)
  - operator^=, [99](#), [100](#)
  - operator+, [89](#)
  - operator+=, [90](#)
  - operator-, [90](#)
  - operator-=, [90](#), [91](#)
  - operator/=: [91](#)
  - operator=, [92d](#)
  - operator%=: [88](#)
  - operator&=: [88](#), [89](#)
  - resize, [100](#)
  - shift, [101](#)
  - size, [101](#)
  - slice, [84](#)
  - slice\_array, [84](#)
  - start, [101](#)
  - stride, [102](#)
  - sum, [102](#)
  - swap, [102](#)
  - valarray, [84](#), [85](#)
- numeric\_traits.h, [3120](#)
- numeric\_fwd.h, [3120](#)
- Numerics, [55](#)
- numpunct
  - std::numpunct, [2597](#)
- oct
  - std, [521](#)
  - std::basic\_fstream, [1616](#)
  - std::basic\_ifstream, [1656](#)
  - std::basic\_ios, [1679](#)
  - std::basic\_iostream, [1726](#)
  - std::basic\_istream, [1764](#)
  - std::basic\_istreamstream, [1805](#)
  - std::basic\_ofstream, [1838](#)
  - std::basic\_ostream, [1869](#)
  - std::basic\_ostreamstream, [1902](#)
  - std::basic\_stringstream, [2038](#)
  - std::ios\_base, [2317](#)
- off\_type
  - std::basic\_ios, [1663](#)
  - std::basic\_streambuf, [1915](#)
- ofstream
  - I/O, [44](#)
- omp\_loop.h, [3122](#)
- omp\_loop\_static.h, [3123](#)
- once\_flag
  - std::once\_flag, [2605](#)
- open
  - \_\_gnu\_cxx::enc\_filebuf, [719](#), [720](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [770](#), [771](#)
  - std::basic\_filebuf, [1555](#), [1556](#)
  - std::basic\_fstream, [1590](#), [1591](#)
  - std::basic\_ifstream, [1637](#)
  - std::basic\_ofstream, [1821](#)
- openmode
  - std::basic\_fstream, [1577](#)
  - std::basic\_ifstream, [1624](#)
  - std::basic\_ios, [1663](#)
  - std::basic\_iostream, [1688](#)
  - std::basic\_istream, [1734](#)
  - std::basic\_istreamstream, [1774](#)
  - std::basic\_ofstream, [1813](#)
  - std::basic\_ostream, [1846](#)
  - std::basic\_ostreamstream, [1878](#)
  - std::basic\_stringstream, [2000](#)
  - std::ios\_base, [2308](#)
- operator\_iterator
  - \_\_gnu\_debug::Safe\_iterator, [820](#)
  - \_\_gnu\_debug::Safe\_local\_iterator, [832](#)

- operator \_RAIter
  - \_\_gnu\_parallel::\_GuardedIterator, 916
- operator bool
  - std::basic\_istream::sentry, 1767
  - std::basic\_ostream::sentry, 1871
  - std::function< \_Res(\_ArgTypes...)>, 2248
  - std::tr2::bool\_set, 2747
- operator const point\_iterator\_
  - iterator\_, 1274
- operator delete
  - new, 3110, 3111
- operator new
  - new, 3112
- operator point\_iterator\_
  - iterator\_, 1274
- operator streamoff
  - std::fpos, 2241
- operator string\_type
  - std::sub\_match, 2714
- operator void \*
  - std::basic\_fstream, 1591
  - std::basic\_ifstream, 1637
  - std::basic\_ios, 1670
  - std::basic\_iostream, 1700
  - std::basic\_istream, 1746
  - std::basic\_istreamstream, 1786
  - std::basic\_ofstream, 1822
  - std::basic\_ostream, 1853
  - std::basic\_ostreamstream, 1886
  - std::basic\_stringstream, 2012
- operator<
  - \_\_gnu\_cxx, 306, 307
  - \_\_gnu\_parallel::\_GuardedIterator, 917
  - Regular Expressions, 190d
  - std, 526d
  - std::tr2, 625
  - Utilities, 69
- operator<<
  - Bernoulli Distributions, 279
  - Complex Numbers, 36
  - Normal Distributions, 277
  - Pointer Abstractions, 51
  - Poisson Distributions, 283, 284
  - Random Number Generators, 270
  - Regular Expressions, 193
  - std, 531d
  - std::basic\_fstream, 1591d
  - std::basic\_iostream, 1701d
  - std::basic\_ofstream, 1822d
  - std::basic\_ostream, 1853d
  - std::basic\_ostreamstream, 1886d
  - std::basic\_stringstream, 2013d
  - std::binomial\_distribution, 2052
  - std::chi\_squared\_distribution, 2063
  - std::discard\_block\_engine, 2197
  - std::discrete\_distribution, 2201
  - std::fisher\_f\_distribution, 2221
  - std::gamma\_distribution, 2255
  - std::linear\_congruential\_engine, 2389
  - std::lognormal\_distribution, 2428
  - std::mersenne\_twister\_engine, 2466
  - std::negative\_binomial\_distribution, 2538
  - std::normal\_distribution, 2543
  - std::piecewise\_constant\_distribution, 2622
  - std::piecewise\_linear\_distribution, 2626
  - std::poisson\_distribution, 2636
  - std::shuffle\_order\_engine, 2702
  - std::student\_t\_distribution, 2710
  - std::tr2, 625
  - std::tr2::dynamic\_bitset, 2756
  - Uniform Distributions, 273
- operator<=<=
  - Numeric Arrays, 91, 92
  - std, 536
  - std::tr2::dynamic\_bitset, 2756
- operator<=
  - \_\_gnu\_cxx, 307, 308
  - \_\_gnu\_parallel::\_GuardedIterator, 917
  - Regular Expressions, 193d
  - std, 536d
  - std::\_\_debug, 570
  - std::\_\_profile, 596
  - std::rel\_ops, 617
  - std::tr2, 625
  - Utilities, 69
- operator>
  - \_\_gnu\_cxx, 310
  - Regular Expressions, 198d
  - std, 544, 545
  - std::\_\_debug, 570
  - std::\_\_profile, 596
  - std::rel\_ops, 617
  - std::tr2, 626
  - Utilities, 70
- operator>>
  - Bernoulli Distributions, 280
  - Complex Numbers, 36
  - Normal Distributions, 277
  - Poisson Distributions, 284, 285
  - std, 548d
  - std::basic\_fstream, 1596d
  - std::basic\_ifstream, 1638d
  - std::basic\_iostream, 1705d
  - std::basic\_istream, 1746d
  - std::basic\_istreamstream, 1786d
  - std::basic\_stringstream, 2017d
  - std::binomial\_distribution, 2052
  - std::chi\_squared\_distribution, 2063

- std::discard\_block\_engine, 2198
- std::discrete\_distribution, 2201
- std::fisher\_f\_distribution, 2221
- std::gamma\_distribution, 2255
- std::independent\_bits\_engine, 2295
- std::linear\_congruential\_engine, 2390
- std::lognormal\_distribution, 2429
- std::mersenne\_twister\_engine, 2467
- std::negative\_binomial\_distribution, 2539
- std::normal\_distribution, 2544
- std::piecewise\_constant\_distribution, 2622
- std::piecewise\_linear\_distribution, 2626
- std::poisson\_distribution, 2636
- std::shuffle\_order\_engine, 2703
- std::student\_t\_distribution, 2710
- std::tr2, 626
- std::tr2::dynamic\_bitset, 2757
- Uniform Distributions, 273, 274
- operator>>=
  - Numeric Arrays, 95, 96
  - std, 553
  - std::tr2::dynamic\_bitset, 2757
- operator>=
  - \_\_gnu\_cxx, 311
  - Regular Expressions, 200d
  - std, 546, 547
  - std::\_\_debug, 570
  - std::\_\_profile, 596
  - std::rel\_ops, 618
  - std::tr2, 626
  - Utilities, 70
- operator\*
  - \_\_gnu\_debug::Safe\_iterator, 820
  - \_\_gnu\_debug::Safe\_local\_iterator, 832
  - \_\_gnu\_parallel::GuardedIterator, 916
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_, 1015
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_, 1021
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 1031
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 1034
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 1075
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 1080
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_, 1094
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer, 1113
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter, 1117
  - Complex Numbers, 33, 34
  - const\_iterator\_, 1267
  - iterator\_, 1274
  - point\_const\_iterator\_, 1278
  - point\_iterator\_, 1280
  - std::auto\_ptr, 1537
  - std::back\_insert\_iterator, 1542
  - std::front\_insert\_iterator, 2244
  - std::insert\_iterator, 2301
  - std::istreambuf\_iterator, 2379
  - std::ostreambuf\_iterator, 2612
  - std::regex\_iterator, 2656
  - std::regex\_token\_iterator, 2659
  - std::reverse\_iterator, 2671
- operator\*=
  - Complex Numbers, 34
  - Numeric Arrays, 89
- operator~
  - Numeric Arrays, 100
  - std, 554
  - std::tr2::dynamic\_bitset, 2758
- operator^
  - std, 554
  - std::tr2, 626
- operator^=
  - Numeric Arrays, 99, 100
  - std::tr2::dynamic\_bitset, 2758
- operator()
  - \_\_gnu\_cxx::subtractive\_rng, 801
  - \_\_gnu\_parallel::Nothing, 947
  - \_\_gnu\_parallel::RandomNumber, 956
  - \_\_gnu\_parallel::\_\_accumulate\_selector, 875
  - \_\_gnu\_parallel::\_\_adjacent\_find\_selector, 877
  - \_\_gnu\_parallel::\_\_count\_if\_selector, 882
  - \_\_gnu\_parallel::\_\_count\_selector, 883
  - \_\_gnu\_parallel::\_\_fill\_selector, 885
  - \_\_gnu\_parallel::\_\_find\_first\_of\_selector, 886
  - \_\_gnu\_parallel::\_\_find\_if\_selector, 888
  - \_\_gnu\_parallel::\_\_for\_each\_selector, 889
  - \_\_gnu\_parallel::\_\_generate\_selector, 890
  - \_\_gnu\_parallel::\_\_identity\_selector, 894
  - \_\_gnu\_parallel::\_\_inner\_product\_selector, 896
  - \_\_gnu\_parallel::\_\_mismatch\_selector, 898
  - \_\_gnu\_parallel::\_\_replace\_if\_selector, 903
  - \_\_gnu\_parallel::\_\_replace\_selector, 905
  - \_\_gnu\_parallel::\_\_transform1\_selector, 906
  - \_\_gnu\_parallel::\_\_transform2\_selector, 907
  - \_\_gnu\_pbds::direct\_mask\_range\_hashing, 1172
  - \_\_gnu\_pbds::direct\_mod\_range\_hashing, 1174
  - \_\_gnu\_pbds::linear\_probe\_fn, 1188
  - \_\_gnu\_pbds::lu\_counter\_policy, 1192
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy, 1193
  - \_\_gnu\_pbds::quadratic\_probe\_fn, 1202
  - \_\_gnu\_pbds::sample\_probe\_fn, 1206
  - \_\_gnu\_pbds::sample\_range\_hashing, 1207
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 1208
  - \_\_gnu\_pbds::sample\_trie\_node\_update, 1217
  - \_\_gnu\_pbds::sample\_update\_policy, 1218

- \_\_gnu\_pbds::tree\_order\_statistics\_node\_update, 1226
- \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, 1232
- \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 1235
- std::bernoulli\_distribution, 2041
- std::binomial\_distribution, 2051
- std::cauchy\_distribution, 2055
- std::chi\_squared\_distribution, 2062
- std::discard\_block\_engine, 2197
- std::discrete\_distribution, 2200
- std::exponential\_distribution, 2213
- std::extreme\_value\_distribution, 2217
- std::fisher\_f\_distribution, 2220
- std::function< \_Res(\_ArgTypes...)>, 2248
- std::gamma\_distribution, 2254
- std::geometric\_distribution, 2257
- std::independent\_bits\_engine, 2294
- std::linear\_congruential\_engine, 2388
- std::locale, 2413
- std::lognormal\_distribution, 2427
- std::negative\_binomial\_distribution, 2537
- std::normal\_distribution, 2542, 2543
- std::piecewise\_constant\_distribution, 2621
- std::piecewise\_linear\_distribution, 2625
- std::poisson\_distribution, 2635
- std::shuffle\_order\_engine, 2701
- std::student\_t\_distribution, 2709
- std::uniform\_int\_distribution, 2776
- std::uniform\_real\_distribution, 2779
- std::weibull\_distribution, 2887
- operator+
  - \_\_gnu\_cxx, 305, 306
  - Complex Numbers, 34
  - Numeric Arrays, 89
  - std, 524d
  - std::fpos, 2241
  - std::reverse\_iterator, 2671
- operator++
  - \_\_gnu\_debug::\_\_Safe\_iterator, 821
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 832, 833
  - \_\_gnu\_parallel::\_\_GuardedIterator, 916
  - const\_iterator\_, 1268
  - iterator\_, 1274, 1275
  - std::back\_insert\_iterator, 1542
  - std::front\_insert\_iterator, 2244
  - std::insert\_iterator, 2301
  - std::istreambuf\_iterator, 2379
  - std::ostreambuf\_iterator, 2612
  - std::regex\_iterator, 2656
  - std::regex\_token\_iterator, 2659, 2660
  - std::reverse\_iterator, 2671
- operator+=
  - \_\_gnu\_cxx::\_\_versa\_string, 680d
  - \_\_gnu\_debug::\_\_basic\_string, 868
  - Complex Numbers, 34
  - Numeric Arrays, 90
  - std::basic\_string, 1962
  - std::complex, 2098
  - std::fpos, 2241
  - std::reverse\_iterator, 2671
- operator-
  - Complex Numbers, 34, 35
  - Numeric Arrays, 90
  - std::fpos, 2241
  - std::reverse\_iterator, 2671
  - std::tr2, 625
- operator->
  - \_\_gnu\_debug::\_\_Safe\_iterator, 821
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 833
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 1031
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 1034
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 1075
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 1080
  - const\_iterator\_, 1268
  - iterator\_, 1275
  - point\_const\_iterator\_, 1278
  - point\_iterator\_, 1281
  - std::auto\_ptr, 1538
  - std::regex\_iterator, 2656
  - std::regex\_token\_iterator, 2660
  - std::reverse\_iterator, 2672
- operator--
  - \_\_gnu\_debug::\_\_Safe\_iterator, 821
  - std::reverse\_iterator, 2672
- operator-=
  - Complex Numbers, 35
  - Numeric Arrays, 90, 91
  - std::complex, 2098
  - std::fpos, 2241
  - std::reverse\_iterator, 2672
  - std::tr2::dynamic\_bitset, 2756
- operator/
  - Complex Numbers, 35
- operator/=
  - Complex Numbers, 35
  - Numeric Arrays, 91
- operator=
  - \_\_gnu\_cxx::\_\_versa\_string, 682, 683
  - \_\_gnu\_debug::\_\_Safe\_iterator, 822
  - \_\_gnu\_debug::\_\_Safe\_local\_iterator, 833
  - Complex Numbers, 36
  - Numeric Arrays, 92d
  - std::auto\_ptr, 1538

- std::back\_insert\_iterator, 1542
- std::basic\_regex, 1911
- std::basic\_string, 1963, 1964
- std::deque, 2189, 2190
- std::forward\_list, 2236
- std::front\_insert\_iterator, 2244
- std::function< \_Res(\_ArgTypes...)>, 2248, 2249
- std::insert\_iterator, 2301
- std::list, 2403
- std::locale, 2413
- std::map, 2443, 2444
- std::match\_results, 2455
- std::multimap, 2516, 2517
- std::multiset, 2532
- std::once\_flag, 2605
- std::ostream\_iterator, 2609
- std::ostreambuf\_iterator, 2612
- std::regex\_iterator, 2656
- std::regex\_token\_iterator, 2660
- std::set, 2690
- std::tr2::dynamic\_bitset, 2756, 2757
- std::unordered\_map, 2801, 2802
- std::unordered\_multimap, 2820, 2821
- std::unordered\_multiset, 2838
- std::unordered\_set, 2857
- std::vector, 2877
- operator==
  - \_\_gnu\_cxx, 308, 309
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator, 1015
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator, 1021
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator, 1031
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator, 1034
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator, 1076
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator, 1080
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 1114
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 1117
- Complex Numbers, 36
- const\_iterator, 1268
- iterator, 1275
- Iterators, 234
- point\_const\_iterator, 1278
- point\_iterator, 1281
- Regular Expressions, 195d
- std, 538d
- std::bernoulli\_distribution, 2042
- std::binomial\_distribution, 2052
- std::cauchy\_distribution, 2055
- std::chi\_squared\_distribution, 2063
- std::discard\_block\_engine, 2198
- std::discrete\_distribution, 2201
- std::exponential\_distribution, 2213
- std::extreme\_value\_distribution, 2218
- std::fisher\_f\_distribution, 2221
- std::gamma\_distribution, 2255
- std::geometric\_distribution, 2258
- std::independent\_bits\_engine, 2295
- std::linear\_congruential\_engine, 2389
- std::locale, 2414
- std::lognormal\_distribution, 2429
- std::mersenne\_twister\_engine, 2466
- std::negative\_binomial\_distribution, 2539
- std::normal\_distribution, 2544
- std::piecewise\_constant\_distribution, 2622
- std::piecewise\_linear\_distribution, 2626
- std::poisson\_distribution, 2636
- std::regex\_iterator, 2656
- std::regex\_token\_iterator, 2660
- std::shuffle\_order\_engine, 2702
- std::student\_t\_distribution, 2710
- std::tr2, 625
- std::uniform\_int\_distribution, 2776
- std::uniform\_real\_distribution, 2780
- std::weibull\_distribution, 2887
- Utilities, 69
- operator%=
  - Numeric Arrays, 88
- operator&
  - std, 524
  - std::tr2, 624
- operator&=
  - Numeric Arrays, 88, 89
  - std::tr2::dynamic\_bitset, 2755, 2756
- opt\_random.h, 3123
- optimize
  - std::regex\_constants, 616
- order\_preserving
  - \_\_gnu\_pbds::container\_traits, 1005
- order\_of\_key
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update, 1226
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, 1232
- order\_of\_prefix
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update, 1232
- order\_statistics\_imp.hpp, 3123
- os\_defines.h, 3124
- ostream, 3124
  - I/O, 44
- ostream.tcc, 3125
- ostream\_insert.h, 3126
- ostream\_iterator

- std::ostream\_iterator, 2608
- ostream\_type
  - std::ostream\_iterator, 2607
  - std::ostreambuf\_iterator, 2611
- ostreambuf\_iterator
  - std::ostreambuf\_iterator, 2611
- ostreamstring
  - I/O, 44
- out
  - std::\_\_codecvt\_abstract\_base, 1289
  - std::basic\_fstream, 1616
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1726
  - std::basic\_istream, 1764
  - std::basic\_istreamstream, 1805
  - std::basic\_ofstream, 1838
  - std::basic\_ostream, 1869
  - std::basic\_ostreamstring, 1902
  - std::basic\_stringstream, 2038
  - std::codecvt, 2070
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2074
  - std::codecvt< char, char, mbstate\_t >, 2078
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2081
  - std::codecvt\_byname, 2086
  - std::ios\_base, 2317
- ov\_tree\_map.hpp, 3126
- overflow
  - \_\_gnu\_cxx::enc\_filebuf, 720
  - \_\_gnu\_cxx::stdio\_filebuf, 771
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 789
  - std::basic\_filebuf, 1556
  - std::basic\_streambuf, 1919
  - std::basic\_stringbuf, 1980
- owner\_less< \_Tp >, 1275
- p
  - std::bernoulli\_distribution, 2041
  - std::binomial\_distribution, 2051
  - std::geometric\_distribution, 2258
  - std::negative\_binomial\_distribution, 2538
- pair
  - std::pair, 2618
- pairing\_heap.hpp, 3127
- par\_loop.h, 3127
- parallel.h, 3127
- parallel\_balanced
  - \_\_gnu\_parallel, 334
- parallel\_omp\_loop
  - \_\_gnu\_parallel, 334
- parallel\_omp\_loop\_static
  - \_\_gnu\_parallel, 334
- parallel\_taskqueue
  - \_\_gnu\_parallel, 334
- parallel\_unbalanced
  - \_\_gnu\_parallel, 334
- parallel\_multiway\_merge
  - \_\_gnu\_parallel, 366
- parallel\_sort\_mwms
  - \_\_gnu\_parallel, 367
- parallel\_sort\_mwms\_pu
  - \_\_gnu\_parallel, 367
- parallel\_tag
  - \_\_gnu\_parallel::parallel\_tag, 982
- param
  - std::bernoulli\_distribution, 2041, 2042
  - std::binomial\_distribution, 2051
  - std::cauchy\_distribution, 2055
  - std::chi\_squared\_distribution, 2062
  - std::discrete\_distribution, 2200
  - std::exponential\_distribution, 2213
  - std::extreme\_value\_distribution, 2217, 2218
  - std::fisher\_f\_distribution, 2220, 2221
  - std::gamma\_distribution, 2254
  - std::geometric\_distribution, 2258
  - std::lognormal\_distribution, 2428
  - std::negative\_binomial\_distribution, 2538
  - std::normal\_distribution, 2543
  - std::piecewise\_constant\_distribution, 2621
  - std::piecewise\_linear\_distribution, 2625, 2626
  - std::poisson\_distribution, 2635, 2636
  - std::student\_t\_distribution, 2709, 2710
  - std::uniform\_int\_distribution, 2776
  - std::uniform\_real\_distribution, 2779
  - std::weibull\_distribution, 2887
- partial\_sort
  - Sorting, 144, 145
- partial\_sort\_copy
  - Sorting, 145, 146
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, 963
- partial\_sum
  - std, 555
- partial\_sum.h, 3128
- partial\_sum\_dilation
  - \_\_gnu\_parallel::Settings, 963
- partial\_sum\_minimal\_n
  - \_\_gnu\_parallel::Settings, 963
- partition
  - Mutating, 110
- partition.h, 3128
  - \_GLIBCXX\_VOLATILE, 3129
- partition\_chunk\_share
  - \_\_gnu\_parallel::Settings, 964
- partition\_chunk\_size
  - \_\_gnu\_parallel::Settings, 964
- partition\_copy

- Mutating, 111
- partition\_minimal\_n
  - \_\_gnu\_parallel::Settings, 964
- partition\_point
  - Mutating, 111
- pat\_trie.hpp, 3129
- pat\_trie\_base.hpp, 3129
- pbackfail
  - \_\_gnu\_cxx::enc\_filebuf, 720
  - \_\_gnu\_cxx::stdio\_filebuf, 772
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 790
  - std::basic\_filebuf, 1557
  - std::basic\_streambuf, 1919
  - std::basic\_stringbuf, 1981
- pbase
  - \_\_gnu\_cxx::enc\_filebuf, 721
  - \_\_gnu\_cxx::stdio\_filebuf, 772
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 790
  - std::basic\_filebuf, 1557
  - std::basic\_streambuf, 1920
  - std::basic\_stringbuf, 1981
- pbump
  - \_\_gnu\_cxx::enc\_filebuf, 721
  - \_\_gnu\_cxx::stdio\_filebuf, 773
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 791
  - std::basic\_filebuf, 1558
  - std::basic\_streambuf, 1920
  - std::basic\_stringbuf, 1981
- peek
  - std::basic\_fstream, 1601
  - std::basic\_ifstream, 1643
  - std::basic\_iostream, 1710
  - std::basic\_istream, 1751
  - std::basic\_istreamstream, 1791
  - std::basic\_stringstream, 2022
- piecewise\_construct
  - Utilities, 71
- pod\_char\_traits.h, 3131
- point\_const\_iterator.hpp, 3131, 3132
- point\_const\_iterator\_, 1276
  - const\_pointer, 1277
  - const\_reference, 1277
  - difference\_type, 1277
  - iterator\_category, 1277
  - operator\*, 1278
  - operator->, 1278
  - operator==, 1278
  - point\_const\_iterator\_, 1277, 1278
  - point\_const\_iterator\_, 1277, 1278
  - pointer, 1277
  - reference, 1277
  - value\_type, 1277
- point\_iterator.hpp, 3132
- point\_iterator\_, 1279
  - const\_pointer, 1279
  - const\_reference, 1279
  - difference\_type, 1279
  - iterator\_category, 1280
  - operator\*, 1280
  - operator->, 1281
  - operator==, 1281
  - point\_iterator\_, 1280
  - point\_iterator\_, 1280
  - pointer, 1280
  - reference, 1280
  - value\_type, 1280
- point\_iterators.hpp, 3132
- pointer
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 1030
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 1033
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 1074
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 1079
  - const\_iterator\_, 1267
  - iterator\_, 1273
  - point\_const\_iterator\_, 1277
  - point\_iterator\_, 1280
  - std::allocator\_traits, 1497
  - std::back\_insert\_iterator, 1541
  - std::front\_insert\_iterator, 2243
  - std::insert\_iterator, 2300
  - std::istream\_iterator, 2374
  - std::istreambuf\_iterator, 2377
  - std::iterator, 2381
  - std::ostream\_iterator, 2608
  - std::ostreambuf\_iterator, 2611
  - std::pointer\_traits, 2632
  - std::pointer\_traits<\_Tp\*>, 2633
  - std::raw\_storage\_iterator, 2651
  - std::set, 2680
  - std::unordered\_map, 2788
  - std::unordered\_multimap, 2808
  - std::unordered\_multiset, 2826
  - std::unordered\_set, 2844
- Pointer Abstractions, 48
  - allocate\_shared, 50
  - get\_deleter, 51
  - make\_shared, 51
  - operator<=, 51
- pointer.h, 3133
- pointer\_to
  - std::pointer\_traits<\_Tp\*>, 2633
- Poisson Distributions, 281
  - operator<=, 283, 284
  - operator>>, 284, 285

- polar
  - Complex Numbers, 36
- Policy-Based Data Structures, 258
- policy\_access\_fnimps.hpp, 3135, 3136
- pool\_allocator.h, 3136
- pop
  - std::priority\_queue, 2639
  - std::queue, 2643
  - std::stack, 2707
- pop\_back
  - \_\_gnu\_cxx::\_\_versa\_string, 684
  - \_\_gnu\_parallel::\_\_RestrictedBoundedConcurrent-  
Queue, 958
  - std::\_\_detail::\_\_Nfa, 1402
  - std::basic\_string, 1964
  - std::deque, 2191
  - std::list, 2404
  - std::vector, 2878
- pop\_front
  - \_\_gnu\_parallel::\_\_RestrictedBoundedConcurrent-  
Queue, 958
  - std::deque, 2191
  - std::forward\_list, 2236
  - std::list, 2404
- pop\_heap
  - Heap, 227, 228
- pos\_format
  - std::moneypunct, 2494
  - std::moneypunct\_byname, 2502
- pos\_type
  - std::basic\_ios, 1663
  - std::basic\_streambuf, 1916
- position
  - std::match\_results, 2456
- positive\_sign
  - std::moneypunct, 2494
  - std::moneypunct\_byname, 2502
- postypes.h, 3137
- pow
  - Complex Numbers, 37
- power
  - SGL, 13
- pptr
  - \_\_gnu\_cxx::enc\_filebuf, 721
  - \_\_gnu\_cxx::stdio\_filebuf, 773
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 791
  - std::basic\_filebuf, 1558
  - std::basic\_streambuf, 1921
  - std::basic\_stringbuf, 1982
- precision
  - std::basic\_fstream, 1601, 1602
  - std::basic\_ifstream, 1643
  - std::basic\_ios, 1670, 1671
  - std::basic\_istream, 1711
  - std::basic\_istream, 1751
  - std::basic\_istream, 1792
  - std::basic\_ofstream, 1827
  - std::basic\_ostream, 1858
  - std::basic\_ostream, 1891
  - std::basic\_stringstream, 2023
  - std::ios\_base, 2310, 2311
- prefix
  - std::match\_results, 2456
- prefix\_range
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 1235,  
1236
- prefix\_search\_node\_update\_imp.hpp, 3137
- prev\_permutation
  - Sorting, 146, 147
- priority\_queue
  - Heap-Based, 256
  - std::priority\_queue, 2639
- priority\_queue.hpp, 3138
- priority\_queue\_base\_dispatch.hpp, 3138
- probabilities
  - std::discrete\_distribution, 2200
- probe\_fn\_base.hpp, 3139
- profiler.h, 3139
- profiler\_algos.h, 3141
- profiler\_container\_size.h, 3142
- profiler\_hash\_func.h, 3142
- profiler\_hashtable\_size.h, 3143
- profiler\_list\_to\_slist.h, 3143
- profiler\_list\_to\_vector.h, 3144
- profiler\_map\_to\_unordered\_map.h, 3144
- profiler\_node.h, 3145
- profiler\_state.h, 3146
- profiler\_trace.h, 3147
- profiler\_vector\_size.h, 3149
- profiler\_vector\_to\_list.h, 3149
- propagate\_on\_container\_copy\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits, 629
  - std::allocator\_traits, 1497
- propagate\_on\_container\_move\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits, 630
  - std::allocator\_traits, 1497
- propagate\_on\_container\_swap
  - \_\_gnu\_cxx::\_\_alloc\_traits, 630
  - std::allocator\_traits, 1497
- ptr\_fun
  - Adaptors for pointers to functions, 221
- ptr\_traits.h, 3150
- pubimbue
  - \_\_gnu\_cxx::enc\_filebuf, 722
  - \_\_gnu\_cxx::stdio\_filebuf, 773
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 791
  - std::basic\_filebuf, 1558
  - std::basic\_streambuf, 1921

- std::basic\_stringbuf, 1982
- pubseekoff
  - \_\_gnu\_cxx::enc\_filebuf, 722
  - \_\_gnu\_cxx::stdio\_filebuf, 773
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 792
  - std::basic\_filebuf, 1559
  - std::basic\_streambuf, 1921
  - std::basic\_stringbuf, 1982
- pubseekpos
  - \_\_gnu\_cxx::enc\_filebuf, 722
  - \_\_gnu\_cxx::stdio\_filebuf, 774
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 792
  - std::basic\_filebuf, 1559
  - std::basic\_streambuf, 1921
  - std::basic\_stringbuf, 1983
- pubsetbuf
  - \_\_gnu\_cxx::enc\_filebuf, 722
  - \_\_gnu\_cxx::stdio\_filebuf, 774
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 792
  - std::basic\_filebuf, 1559
  - std::basic\_streambuf, 1922
  - std::basic\_stringbuf, 1983
- pubsync
  - \_\_gnu\_cxx::enc\_filebuf, 723
  - \_\_gnu\_cxx::stdio\_filebuf, 774
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 792
  - std::basic\_filebuf, 1559
  - std::basic\_streambuf, 1922
  - std::basic\_stringbuf, 1983
- push
  - std::priority\_queue, 2640
  - std::queue, 2643
  - std::stack, 2707
- push\_back
  - \_\_gnu\_cxx::\_\_versa\_string, 684
  - std::\_\_detail::\_\_Nfa, 1402
  - std::basic\_string, 1965
  - std::deque, 2191
  - std::list, 2404
  - std::tr2::dynamic\_bitset, 2758
  - std::vector, 2879
- push\_front
  - \_\_gnu\_parallel::\_\_RestrictedBoundedConcurrent-  
Queue, 958
  - std::deque, 2191
  - std::forward\_list, 2236
  - std::list, 2404
- push\_heap
  - Heap, 228
- put
  - std::basic\_fstream, 1602
  - std::basic\_istream, 1711
  - std::basic\_ofstream, 1828
  - std::basic\_ostream, 1859
  - std::basic\_ostringstream, 1892
  - std::basic\_stringstream, 2023
  - std::money\_put, 2485
  - std::num\_put, 2566d
  - std::time\_put, 2738
  - std::time\_put\_byname, 2741
- put\_money
  - std, 555
- putback
  - std::basic\_fstream, 1602
  - std::basic\_ifstream, 1644
  - std::basic\_iostream, 1712
  - std::basic\_istream, 1752
  - std::basic\_istringstream, 1792
  - std::basic\_stringstream, 2024
- pword
  - std::basic\_fstream, 1603
  - std::basic\_ifstream, 1644
  - std::basic\_ios, 1671
  - std::basic\_iostream, 1712
  - std::basic\_istream, 1752
  - std::basic\_istringstream, 1793
  - std::basic\_ofstream, 1828
  - std::basic\_ostream, 1859
  - std::basic\_ostringstream, 1892
  - std::basic\_stringstream, 2024
  - std::ios\_base, 2311
- qsb\_steals
  - \_\_gnu\_parallel::\_\_Settings, 964
- quadratic\_probe\_fn\_imp.hpp, 3150
- queue, 3150
  - std::queue, 2642
- queue.h, 3151
  - \_GLIBCXX\_VOLATILE, 3151
- quicksort.h, 3151
- quiet\_NaN
  - std::numeric\_limits, 2573
- r\_erase\_fn\_imps.hpp, 3152
- radix
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2575
- random, 3152
- Random Number Distributions, 271
- Random Number Generation, 181
  - generate\_canonical, 181
- Random Number Generators, 266
  - minstd\_rand, 267
  - minstd\_rand0, 268
  - mt19937, 268
  - mt19937\_64, 268
  - operator<<, 270
- Random Number Utilities, 286
- random.h, 3152

- random.tcc, [3158](#), [3162](#)
- random\_number.h, [3164](#)
- random\_sample
  - SGL, [13](#), [14](#)
- random\_sample\_n
  - SGL, [14](#)
- random\_shuffle
  - Mutating, [112](#)
- random\_shuffle.h, [3165](#)
- random\_shuffle\_minimal\_n
  - \_\_gnu\_parallel::Settings, [964](#)
- range\_access.h, [3165](#)
- ranged\_hash\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, Store\_Hash >, [1281](#)
- ranged\_hash\_fn.hpp, [3166](#)
- ranged\_probe\_fn.hpp, [3167](#)
- ratio, [3167](#), [3168](#)
- ratio\_divide
  - Rational Arithmetic, [57](#)
- ratio\_multiply
  - Rational Arithmetic, [57](#)
- Rational Arithmetic, [56](#)
  - ratio\_divide, [57](#)
  - ratio\_multiply, [57](#)
- rb\_tree, [3168](#)
- rb\_tree\_.hpp, [3169](#)
- rbegin
  - \_\_gnu\_cxx::\_\_versa\_string, [685](#)
  - std::\_\_detail::\_Nfa, [1403](#)
  - std::basic\_string, [1965](#)
  - std::deque, [2192](#)
  - std::list, [2404](#), [2405](#)
  - std::map, [2444](#), [2445](#)
  - std::multimap, [2517](#)
  - std::multiset, [2532](#)
  - std::set, [2690](#)
  - std::vector, [2879](#)
- rc.hpp, [3169](#)
- rc\_binomial\_heap\_.hpp, [3169](#)
- rc\_string\_base.h, [3170](#)
- rdbuf
  - std::basic\_fstream, [1603](#), [1604](#)
  - std::basic\_ifstream, [1645](#)
  - std::basic\_ios, [1671](#), [1672](#)
  - std::basic\_iostream, [1713](#)
  - std::basic\_istream, [1753](#)
  - std::basic\_istreamstream, [1793](#), [1794](#)
  - std::basic\_ofstream, [1829](#)
  - std::basic\_ostream, [1859](#), [1860](#)
  - std::basic\_ostreamstream, [1892](#), [1893](#)
  - std::basic\_stringstream, [2025](#)
- rdstate
  - std::basic\_fstream, [1604](#)
  - std::basic\_ifstream, [1645](#)
- std::basic\_ios, [1672](#)
- std::basic\_iostream, [1713](#)
- std::basic\_istream, [1753](#)
- std::basic\_istreamstream, [1794](#)
- std::basic\_ofstream, [1829](#)
- std::basic\_ostream, [1860](#)
- std::basic\_ostreamstream, [1893](#)
- std::basic\_stringstream, [2025](#)
- read
  - std::basic\_fstream, [1604](#)
  - std::basic\_ifstream, [1645](#)
  - std::basic\_iostream, [1714](#)
  - std::basic\_istream, [1754](#)
  - std::basic\_istreamstream, [1794](#)
  - std::basic\_stringstream, [2026](#)
- readsome
  - std::basic\_fstream, [1605](#)
  - std::basic\_ifstream, [1646](#)
  - std::basic\_iostream, [1714](#)
  - std::basic\_istream, [1754](#)
  - std::basic\_istreamstream, [1795](#)
  - std::basic\_stringstream, [2026](#)
- ready
  - std::match\_results, [2456](#)
- ref
  - std, [556](#)
- reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator\_, [1014](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator\_, [1020](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, [1030](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, [1033](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, [1074](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, [1079](#)
  - const\_iterator\_, [1267](#)
  - iterator\_, [1273](#)
  - point\_const\_iterator\_, [1277](#)
  - point\_iterator\_, [1280](#)
  - std::back\_insert\_iterator, [1541](#)
  - std::front\_insert\_iterator, [2243](#)
  - std::insert\_iterator, [2300](#)
  - std::istream\_iterator, [2374](#)
  - std::istreambuf\_iterator, [2377](#)
  - std::iterator, [2381](#)
  - std::ostream\_iterator, [2608](#)
  - std::ostreambuf\_iterator, [2611](#)
  - std::raw\_storage\_iterator, [2651](#)
  - std::set, [2680](#)
  - std::unordered\_map, [2788](#)
  - std::unordered\_multimap, [2808](#)

```
__gnu_cxx::temporary_buffer, 803
std:: Temporary buffer, 1481
```

- reserve
  - `__gnu_cxx::__versa_string`, [691](#)
  - `__gnu_debug::basic_string`, [872](#)
  - `std::__detail::_Nfa`, [1403](#)
  - `std::basic_string`, [1971](#)
  - `std::unordered_map`, [2803](#)
  - `std::unordered_multimap`, [2821](#)
  - `std::unordered_multiset`, [2839](#)
  - `std::unordered_set`, [2857](#)
  - `std::vector`, [2879](#)
- reset
  - `std`, [556](#)
  - `std::auto_ptr`, [1539](#)
  - `std::bernoulli_distribution`, [2042](#)
  - `std::binomial_distribution`, [2052](#)
  - `std::cauchy_distribution`, [2055](#)
  - `std::chi_squared_distribution`, [2062](#)
  - `std::discrete_distribution`, [2201](#)
  - `std::exponential_distribution`, [2213](#)
  - `std::extreme_value_distribution`, [2218](#)
  - `std::fisher_f_distribution`, [2221](#)
  - `std::gamma_distribution`, [2254](#)
  - `std::geometric_distribution`, [2258](#)
  - `std::lognormal_distribution`, [2428](#)
  - `std::negative_binomial_distribution`, [2538](#)
  - `std::normal_distribution`, [2543](#)
  - `std::piecewise_constant_distribution`, [2621](#)
  - `std::piecewise_linear_distribution`, [2626](#)
  - `std::poisson_distribution`, [2636](#)
  - `std::student_t_distribution`, [2710](#)
  - `std::tr2::dynamic_bitset`, [2759](#)
  - `std::uniform_int_distribution`, [2776](#)
  - `std::uniform_real_distribution`, [2779](#)
  - `std::weibull_distribution`, [2887](#)
- resetiosflags
  - `std`, [557](#)
- resize
  - `__gnu_cxx::__versa_string`, [691](#), [692](#)
  - `__gnu_pbds::hash_standard_resize_policy`, [1186](#)
  - Numeric Arrays, [100](#)
  - `std::__detail::_Nfa`, [1403](#), [1404](#)
  - `std::basic_string`, [1971](#), [1972](#)
  - `std::deque`, [2192](#), [2193](#)
  - `std::forward_list`, [2237](#), [2238](#)
  - `std::list`, [2406](#)
  - `std::tr2::dynamic_bitset`, [2759](#)
  - `std::vector`, [2880](#)
- `resize_fn_imps.hpp`, [3181](#)
- `resize_no_store_hash_fn_imps.hpp`, [3181](#)
- `resize_policy.hpp`, [3182](#)
- `resize_store_hash_fn_imps.hpp`, [3182](#)
- `result_of< _Signature >`, [1282](#)
- result\_type
  - `__gnu_cxx::__detail::_Ffit_finder`, [636](#)
  - `__gnu_cxx::binary_compose`, [706](#)
  - `__gnu_cxx::project1st`, [746](#)
  - `__gnu_cxx::project2nd`, [747](#)
  - `__gnu_cxx::select1st`, [759](#)
  - `__gnu_cxx::select2nd`, [760](#)
  - `__gnu_cxx::subtractive_rng`, [801](#)
  - `__gnu_cxx::unary_compose`, [812](#)
  - `__gnu_parallel::_EqualFromLess`, [914](#)
  - `__gnu_parallel::_EqualTo`, [915](#)
  - `__gnu_parallel::_Less`, [922](#)
  - `__gnu_parallel::_Lexicographic`, [924](#)
  - `__gnu_parallel::_LexicographicReverse`, [925](#)
  - `__gnu_parallel::_Multiplies`, [947](#)
  - `__gnu_parallel::_Plus`, [949](#)
  - `__gnu_parallel::_binder1st`, [879](#)
  - `__gnu_parallel::_binder2nd`, [881](#)
  - `__gnu_parallel::_unary_negate`, [909](#)
  - `std::_Maybe_unary_or_binary_function< _Res, _T1 >`, [1468](#)
  - `std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`, [1469](#)
  - `std::bernoulli_distribution`, [2041](#)
  - `std::binary_function`, [2045](#)
  - `std::binary_negate`, [2046](#)
  - `std::binder1st`, [2048](#)
  - `std::binder2nd`, [2049](#)
  - `std::binomial_distribution`, [2051](#)
  - `std::cauchy_distribution`, [2054](#)
  - `std::chi_squared_distribution`, [2061](#)
  - `std::const_mem_fun1_ref_t`, [2103](#)
  - `std::const_mem_fun1_t`, [2105](#)
  - `std::const_mem_fun_ref_t`, [2106](#)
  - `std::const_mem_fun_t`, [2108](#)
  - `std::discard_block_engine`, [2195](#)
  - `std::discrete_distribution`, [2200](#)
  - `std::divides`, [2203](#)
  - `std::equal_to`, [2207](#)
  - `std::exponential_distribution`, [2212](#)
  - `std::extreme_value_distribution`, [2217](#)
  - `std::fisher_f_distribution`, [2220](#)
  - `std::gamma_distribution`, [2253](#)
  - `std::geometric_distribution`, [2257](#)
  - `std::greater`, [2260](#)
  - `std::greater_equal`, [2262](#)
  - `std::hash< __gnu_cxx::throw_value_limit >`, [2273](#)
  - `std::hash< __gnu_cxx::throw_value_random >`, [2275](#)
  - `std::independent_bits_engine`, [2292](#)
  - `std::less`, [2384](#)
  - `std::less_equal`, [2386](#)
  - `std::linear_congruential_engine`, [2387](#)
  - `std::logical_and`, [2423](#)
  - `std::logical_not`, [2424](#)
  - `std::logical_or`, [2425](#)

- std::lognormal\_distribution, 2427
- std::mem\_fun1\_ref\_t, 2459
- std::mem\_fun1\_t, 2460
- std::mem\_fun\_ref\_t, 2462
- std::mem\_fun\_t, 2463
- std::mersenne\_twister\_engine, 2465
- std::minus, 2474
- std::modulus, 2476
- std::multiplies, 2520
- std::negate, 2535
- std::negative\_binomial\_distribution, 2537
- std::normal\_distribution, 2542
- std::not\_equal\_to, 2546
- std::owner\_less< shared\_ptr< \_Tp > >, 2615
- std::owner\_less< weak\_ptr< \_Tp > >, 2616
- std::piecewise\_constant\_distribution, 2620
- std::piecewise\_linear\_distribution, 2625
- std::plus, 2628
- std::pointer\_to\_binary\_function, 2630
- std::pointer\_to\_unary\_function, 2631
- std::poisson\_distribution, 2635
- std::random\_device, 2645
- std::seed\_seq, 2676
- std::shuffle\_order\_engine, 2700
- std::student\_t\_distribution, 2709
- std::unary\_function, 2771
- std::unary\_negate, 2773
- std::uniform\_int\_distribution, 2775
- std::uniform\_real\_distribution, 2778
- std::weibull\_distribution, 2886
- rethrow\_exception
  - Exceptions, 25
- rethrow\_if\_nested
  - Exceptions, 25, 26
- return\_temporary\_buffer
  - std, 557
- reverse
  - Mutating, 115
  - std::forward\_list, 2238
  - std::list, 2406
- reverse\_iteration
  - \_\_gnu\_pbds::container\_traits, 1006
- reverse\_copy
  - Mutating, 115
- reverse\_iterator
  - std::reverse\_iterator, 2670
  - std::set, 2681
- rfind
  - \_\_gnu\_cxx::\_\_versa\_string, 692, 693
  - \_\_gnu\_debug::basic\_string, 872
  - std::basic\_string, 1972, 1973
- riemann\_zeta
  - Mathematical Special Functions, 245
- right
  - std, 557
  - std::basic\_fstream, 1616
  - std::basic\_ifstream, 1656
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1726
  - std::basic\_istream, 1765
  - std::basic\_istreamstream, 1805
  - std::basic\_ofstream, 1838
  - std::basic\_ostream, 1869
  - std::basic\_ostreamstream, 1903
  - std::basic\_stringstream, 2038
  - std::ios\_base, 2317
- rope, 3182
- ropeimpl.h, 3186
- rotate
  - Mutating, 116
- rotate\_copy
  - Mutating, 116
- rotate\_fn\_imps.hpp, 3186
- round\_to\_nearest
  - std, 498
- round\_toward\_infinity
  - std, 498
- round\_toward\_neg\_infinity
  - std, 498
- round\_toward\_zero
  - std, 498
- round\_error
  - std::numeric\_limits, 2573
- round\_style
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2575
- runtime\_error
  - std::runtime\_error, 2673
- SGL, 7
  - \_Find\_first, 10
  - \_Find\_next, 10
  - \_Unchecked\_flip, 11
  - \_Unchecked\_reset, 11
  - \_Unchecked\_set, 11
  - \_Unchecked\_test, 11
  - \_\_median, 10
  - compose1, 11
  - compose2, 11
  - constant0, 12
  - constant1, 12
  - constant2, 12
  - copy\_n, 12
  - distance, 12
  - identity\_element, 13
  - lexicographical\_compare\_3way, 13
  - power, 13
  - random\_sample, 13, 14

- random\_sample\_n, 14
- uninitialized\_copy\_n, 14
- safe\_base.h, 3186
- safe\_iterator.h, 3187
- safe\_iterator.tcc, 3189
- safe\_local\_iterator.h, 3189
- safe\_local\_iterator.tcc, 3190
- safe\_sequence.h, 3190
- safe\_sequence.tcc, 3190
- safe\_unordered\_base.h, 3191
- safe\_unordered\_container.h, 3191
- safe\_unordered\_container.tcc, 3192
- sample\_probe\_fn
  - \_\_gnu\_pbds::sample\_probe\_fn, 1206
- sample\_probe\_fn.hpp, 3192
- sample\_range\_hashing
  - \_\_gnu\_pbds::sample\_range\_hashing, 1207
  - \_\_gnu\_pbds::sample\_resize\_policy, 1211
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1214
  - \_\_gnu\_pbds::sample\_size\_policy, 1215
- sample\_range\_hashing.hpp, 3192
- sample\_ranged\_hash\_fn
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 1208
- sample\_ranged\_hash\_fn.hpp, 3193
- sample\_ranged\_probe\_fn.hpp, 3193
- sample\_resize\_policy
  - \_\_gnu\_pbds::sample\_resize\_policy, 1210
- sample\_resize\_policy.hpp, 3193
- sample\_resize\_trigger
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1212
- sample\_resize\_trigger.hpp, 3194
- sample\_size\_policy
  - \_\_gnu\_pbds::sample\_size\_policy, 1214
- sample\_size\_policy.hpp, 3194
- sample\_tree\_node\_update.hpp, 3194
- sample\_trie\_access\_traits.hpp, 3195
- sample\_trie\_node\_update
  - \_\_gnu\_pbds::sample\_trie\_node\_update, 1217
- sample\_trie\_node\_update.hpp, 3195
- sample\_update\_policy
  - \_\_gnu\_pbds::sample\_update\_policy, 1218
- sample\_update\_policy.hpp, 3195
- sbumpc
  - \_\_gnu\_cxx::enc\_filebuf, 723
  - \_\_gnu\_cxx::stdio\_filebuf, 774
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 792
  - std::basic\_filebuf, 1559
  - std::basic\_streambuf, 1922
  - std::basic\_stringbuf, 1983
- scan\_is
  - std::\_\_ctype\_abstract\_base, 1299
  - std::ctype, 2116
  - std::ctype< char >, 2127
  - std::ctype< wchar\_t >, 2140
- std::ctype\_byname, 2152
- std::ctype\_byname< char >, 2162
- scan\_not
  - std::\_\_ctype\_abstract\_base, 1300
  - std::ctype, 2117
  - std::ctype< char >, 2127
  - std::ctype< wchar\_t >, 2140
  - std::ctype\_byname, 2152
  - std::ctype\_byname< char >, 2163
- scientific
  - std, 557
  - std::basic\_fstream, 1617
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1726
  - std::basic\_istream, 1765
  - std::basic\_istreamstringstream, 1806
  - std::basic\_ofstream, 1838
  - std::basic\_ostream, 1869
  - std::basic\_ostreamstringstream, 1903
  - std::basic\_stringstream, 2039
  - std::ios\_base, 2317
- scoped\_allocator, 3196
- search
  - Non-Mutating, 130
- search.h, 3196
- search\_minimal\_n
  - \_\_gnu\_parallel::Settings, 964
- search\_n
  - Non-Mutating, 131
- second
  - \_\_gnu\_parallel::IteratorPair, 919
  - std::pair, 2619
  - std::sub\_match, 2715
- second\_argument\_type
  - \_\_gnu\_cxx::project1st, 747
  - \_\_gnu\_cxx::project2nd, 747
  - \_\_gnu\_parallel::EqualFromLess, 914
  - \_\_gnu\_parallel::EqualTo, 915
  - \_\_gnu\_parallel::Less, 923
  - \_\_gnu\_parallel::Lexicographic, 924
  - \_\_gnu\_parallel::LexicographicReverse, 926
  - \_\_gnu\_parallel::Multiplies, 947
  - \_\_gnu\_parallel::Plus, 950
  - std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 >, 1470
  - std::binary\_function, 2045
  - std::binary\_negate, 2046
  - std::const\_mem\_fun1\_ref\_t, 2104
  - std::const\_mem\_fun1\_t, 2105
  - std::divides, 2204
  - std::equal\_to, 2207
  - std::greater, 2261
  - std::greater\_equal, 2262

- std::less, 2385
- std::less\_equal, 2386
- std::logical\_and, 2423
- std::logical\_or, 2426
- std::mem\_fun1\_ref\_t, 2459
- std::mem\_fun1\_t, 2461
- std::minus, 2475
- std::modulus, 2476
- std::multiplies, 2520
- std::not\_equal\_to, 2546
- std::owner\_less< shared\_ptr< \_Tp > >, 2615
- std::owner\_less< weak\_ptr< \_Tp > >, 2616
- std::plus, 2629
- std::pointer\_to\_binary\_function, 2630
- second\_type
  - \_\_gnu\_parallel::\_iteratorPair, 919
  - std::pair, 2618
  - std::sub\_match, 2713
- seconds
  - std::chrono, 600
- seed
  - std::discard\_block\_engine, 2197
  - std::independent\_bits\_engine, 2294
  - std::linear\_congruential\_engine, 2388, 2389
  - std::shuffle\_order\_engine, 2701, 2702
- seed\_seq
  - std::seed\_seq, 2676
- seekdir
  - std::basic\_fstream, 1578
  - std::basic\_ifstream, 1624
  - std::basic\_ios, 1664
  - std::basic\_iostream, 1688
  - std::basic\_istream, 1734
  - std::basic\_istreamstream, 1774
  - std::basic\_ofstream, 1813
  - std::basic\_ostream, 1846
  - std::basic\_ostreamstream, 1878
  - std::basic\_stringstream, 2000
  - std::ios\_base, 2308
- seekg
  - std::basic\_fstream, 1606
  - std::basic\_ifstream, 1647
  - std::basic\_iostream, 1715
  - std::basic\_istream, 1755
  - std::basic\_istreamstream, 1795, 1796
  - std::basic\_stringstream, 2027
- seekoff
  - \_\_gnu\_cxx::enc\_filebuf, 723
  - \_\_gnu\_cxx::stdio\_filebuf, 775
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 793
  - std::basic\_filebuf, 1560
  - std::basic\_streambuf, 1922
  - std::basic\_stringbuf, 1983
- seekp
  - std::basic\_fstream, 1606, 1607
  - std::basic\_iostream, 1716
  - std::basic\_ofstream, 1830
  - std::basic\_ostream, 1861
  - std::basic\_ostreamstream, 1894
  - std::basic\_stringstream, 2028
- seekpos
  - \_\_gnu\_cxx::enc\_filebuf, 723
  - \_\_gnu\_cxx::stdio\_filebuf, 775
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 793
  - std::basic\_filebuf, 1560
  - std::basic\_streambuf, 1923
  - std::basic\_stringbuf, 1984
- select\_on\_container\_copy\_construction
  - \_\_gnu\_cxx::\_\_alloc\_traits, 632
  - std::allocator\_traits, 1499
- sentry
  - std::basic\_istream::sentry, 1767
  - std::basic\_ostream::sentry, 1871
- Sequences, 17
- sequential
  - \_\_gnu\_parallel, 334
- set, 3197
  - \_\_gnu\_parallel::\_Settings, 961
  - std, 557
  - std::set, 2681, 2682
  - std::tr2::dynamic\_bitset, 2759
- Set Operation, 150
  - includes, 151
  - set\_difference, 151, 152
  - set\_intersection, 152, 153
  - set\_symmetric\_difference, 153, 154
  - set\_union, 154
- set.h, 3197, 3198
- set\_difference
  - Set Operation, 151, 152
- set\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 964
- set\_intersection
  - Set Operation, 152, 153
- set\_intersection\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 964
- set\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger, 997
- set\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger, 1182
- set\_new\_handler
  - std, 558
- set\_num\_threads
  - \_\_gnu\_parallel::balanced\_quicksort\_tag, 968
  - \_\_gnu\_parallel::balanced\_tag, 969
  - \_\_gnu\_parallel::default\_parallel\_tag, 970
  - \_\_gnu\_parallel::exact\_tag, 972

- \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 975
  - \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 976
  - \_\_gnu\_parallel::multiway\_mergesort\_tag, 977
  - \_\_gnu\_parallel::omp\_loop\_static\_tag, 979
  - \_\_gnu\_parallel::omp\_loop\_tag, 980
  - \_\_gnu\_parallel::parallel\_tag, 982
  - \_\_gnu\_parallel::quicksort\_tag, 983
  - \_\_gnu\_parallel::sampling\_tag, 984
  - \_\_gnu\_parallel::unbalanced\_tag, 986
- set\_operations.h, 3199
- set\_symmetric\_difference
  - Set Operation, 153, 154
- set\_symmetric\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 964
- set\_terminate
  - Exceptions, 26
- set\_unexpected
  - Exceptions, 26
- set\_union
  - Set Operation, 154
- set\_union\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 965
- setbase
  - std, 558
- setbuf
  - \_\_gnu\_cxx::enc\_filebuf, 723
  - \_\_gnu\_cxx::stdio\_filebuf, 775
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 793
  - std::basic\_filebuf, 1560
  - std::basic\_streambuf, 1923
  - std::basic\_stringbuf, 1984
- setf
  - std::basic\_fstream, 1607
  - std::basic\_ifstream, 1648
  - std::basic\_ios, 1673
  - std::basic\_iostream, 1717
  - std::basic\_istream, 1756
  - std::basic\_istreamstream, 1796, 1797
  - std::basic\_ofstream, 1830, 1831
  - std::basic\_ostream, 1861, 1862
  - std::basic\_ostreamstream, 1894, 1895
  - std::basic\_stringstream, 2029
  - std::ios\_base, 2311, 2312
- setfill
  - std, 558
- setg
  - \_\_gnu\_cxx::enc\_filebuf, 724
  - \_\_gnu\_cxx::stdio\_filebuf, 776
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 794
  - std::basic\_filebuf, 1561
  - std::basic\_streambuf, 1923
  - std::basic\_stringbuf, 1984
- setiosflags
  - std, 558
- setp
  - \_\_gnu\_cxx::enc\_filebuf, 724
  - \_\_gnu\_cxx::stdio\_filebuf, 776
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 794
  - std::basic\_filebuf, 1561
  - std::basic\_streambuf, 1924
  - std::basic\_stringbuf, 1985
- setprecision
  - std, 558
- setstate
  - std::basic\_fstream, 1608
  - std::basic\_ifstream, 1648
  - std::basic\_ios, 1673
  - std::basic\_iostream, 1717
  - std::basic\_istream, 1757
  - std::basic\_istreamstream, 1797
  - std::basic\_ofstream, 1831
  - std::basic\_ostream, 1862
  - std::basic\_ostreamstream, 1895
  - std::basic\_stringstream, 2029
- settings.h, 3200
- setw
  - std, 559
- sgetc
  - \_\_gnu\_cxx::enc\_filebuf, 724
  - \_\_gnu\_cxx::stdio\_filebuf, 776
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 794
  - std::basic\_filebuf, 1562
  - std::basic\_streambuf, 1924
  - std::basic\_stringbuf, 1985
- sgetn
  - \_\_gnu\_cxx::enc\_filebuf, 725
  - \_\_gnu\_cxx::stdio\_filebuf, 777
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 795
  - std::basic\_filebuf, 1562
  - std::basic\_streambuf, 1924
  - std::basic\_stringbuf, 1985
- shared\_ptr
  - std::shared\_ptr, 2694d
- shared\_ptr.h, 3201
- shared\_ptr\_base.h, 3203
- shift
  - Numeric Arrays, 101
- showbase
  - std, 559
  - std::basic\_fstream, 1617
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1727
  - std::basic\_istream, 1765
  - std::basic\_istreamstream, 1806
  - std::basic\_ofstream, 1839
  - std::basic\_ostream, 1870

- std::basic\_ostringstream, 1903
- std::basic\_stringstream, 2039
- std::ios\_base, 2317
- showmanyc
  - \_\_gnu\_cxx::enc\_filebuf, 725
  - \_\_gnu\_cxx::stdio\_filebuf, 777
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 795
  - std::basic\_filebuf, 1562
  - std::basic\_streambuf, 1925
  - std::basic\_stringbuf, 1986
- showpoint
  - std, 559
  - std::basic\_fstream, 1617
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1727
  - std::basic\_istream, 1765
  - std::basic\_istreamstream, 1806
  - std::basic\_ofstream, 1839
  - std::basic\_ostream, 1870
  - std::basic\_ostringstream, 1903
  - std::basic\_stringstream, 2039
  - std::ios\_base, 2317
- showpos
  - std, 559
  - std::basic\_fstream, 1617
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1727
  - std::basic\_istream, 1765
  - std::basic\_istreamstream, 1806
  - std::basic\_ofstream, 1839
  - std::basic\_ostream, 1870
  - std::basic\_ostringstream, 1903
  - std::basic\_stringstream, 2039
  - std::ios\_base, 2318
- shrink\_to\_fit
  - \_\_gnu\_cxx::\_\_versa\_string, 694
  - std::\_\_detail::\_Nfa, 1404
  - std::basic\_string, 1973
  - std::deque, 2193
  - std::vector, 2880
- shuffle
  - Mutating, 117
- shuffle\_order\_engine
  - std::shuffle\_order\_engine, 2700
- signaling\_NaN
  - std::numeric\_limits, 2573
- sin
  - Complex Numbers, 37
- sinh
  - Complex Numbers, 37
- size
  - \_\_gnu\_cxx::\_\_versa\_string, 694
  - \_\_gnu\_cxx::temporary\_buffer, 804
  - \_\_gnu\_debug::basic\_string, 872
  - Numeric Arrays, 101
  - std, 559
  - std::\_\_detail::\_Nfa, 1404
  - std::\_Temporary\_buffer, 1481
  - std::basic\_string, 1973
  - std::deque, 2193
  - std::list, 2406
  - std::map, 2445
  - std::match\_results, 2457
  - std::multimap, 2518
  - std::multiset, 2533
  - std::priority\_queue, 2640
  - std::queue, 2643
  - std::set, 2691
  - std::stack, 2707
  - std::tr2::dynamic\_bitset, 2760
  - std::unordered\_map, 2803
  - std::unordered\_multimap, 2821
  - std::unordered\_multiset, 2839
  - std::unordered\_set, 2858
  - std::vector, 2880
- size\_fn\_imps.hpp, 3205
- size\_type
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, 1183
  - \_\_gnu\_pbds::sample\_range\_hashing, 1207
  - \_\_gnu\_pbds::sample\_resize\_policy, 1210
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1212
  - \_\_gnu\_pbds::sample\_size\_policy, 1214
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update, 1235
  - std::allocator\_traits, 1497
  - std::set, 2681
  - std::unordered\_map, 2788
  - std::unordered\_multimap, 2808
  - std::unordered\_multiset, 2826
  - std::unordered\_set, 2844
- skipws
  - std, 560
  - std::basic\_fstream, 1617
  - std::basic\_ifstream, 1657
  - std::basic\_ios, 1680
  - std::basic\_iostream, 1727
  - std::basic\_istream, 1765
  - std::basic\_istreamstream, 1806
  - std::basic\_ofstream, 1839
  - std::basic\_ostream, 1870
  - std::basic\_ostringstream, 1903
  - std::basic\_stringstream, 2039
  - std::ios\_base, 2318
- sleep\_for
  - std::this\_thread, 618
- sleep\_until
  - std::this\_thread, 619

- slice
  - Numeric Arrays, 84
- slice\_array
  - Numeric Arrays, 84
- slice\_array.h, 3205
- slist, 3205
- snextc
  - \_\_gnu\_cxx::enc\_filebuf, 725
  - \_\_gnu\_cxx::stdio\_filebuf, 777
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 795
  - std::basic\_filebuf, 1563
  - std::basic\_streambuf, 1925
  - std::basic\_stringbuf, 1986
- sort
  - Sorting, 147
  - std::forward\_list, 2238
  - std::list, 2406
- sort.h, 3206
- sort\_heap
  - Heap, 229
- sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, 965
- sort\_mwms\_oversampling
  - \_\_gnu\_parallel::Settings, 965
- sort\_qs\_num\_samples\_preset
  - \_\_gnu\_parallel::Settings, 965
- sort\_qsb\_base\_case\_maximal\_n
  - \_\_gnu\_parallel::Settings, 965
- Sorting, 133
  - inplace\_merge, 135
  - is\_sorted, 136
  - is\_sorted\_until, 136, 137
  - lexicographical\_compare, 137
  - max, 138
  - max\_element, 139
  - merge, 139, 140
  - min, 140, 141
  - min\_element, 141
  - minmax, 142
  - minmax\_element, 142
  - next\_permutation, 143
  - nth\_element, 144
  - partial\_sort, 144, 145
  - partial\_sort\_copy, 145, 146
  - prev\_permutation, 146, 147
  - sort, 147
  - stable\_sort, 148
- sph\_bessel
  - Mathematical Special Functions, 245
- sph\_legendre
  - Mathematical Special Functions, 245
- sph\_neumann
  - Mathematical Special Functions, 245
- splay\_fn\_imps.hpp, 3207
- splay\_tree\_.hpp, 3207
- splice
  - std::list, 2407
- splice\_after
  - std::forward\_list, 2238, 2239
- split\_join\_can\_throw
  - \_\_gnu\_pbds::container\_traits, 1006
- split\_fn\_imps.hpp, 3208
- split\_join\_fn\_imps.hpp, 3208, 3209
- sputbackc
  - \_\_gnu\_cxx::enc\_filebuf, 725
  - \_\_gnu\_cxx::stdio\_filebuf, 778
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 796
  - std::basic\_filebuf, 1563
  - std::basic\_streambuf, 1925
  - std::basic\_stringbuf, 1986
- sputc
  - \_\_gnu\_cxx::enc\_filebuf, 726
  - \_\_gnu\_cxx::stdio\_filebuf, 778
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 796
  - std::basic\_filebuf, 1563
  - std::basic\_streambuf, 1926
  - std::basic\_stringbuf, 1987
- sputn
  - \_\_gnu\_cxx::enc\_filebuf, 726
  - \_\_gnu\_cxx::stdio\_filebuf, 778
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 796
  - std::basic\_filebuf, 1564
  - std::basic\_streambuf, 1926
  - std::basic\_stringbuf, 1987
- sqrt
  - Complex Numbers, 37
- sregex\_token\_iterator
  - Regular Expressions, 187
- sso\_string\_base.h, 3209
- sstream, 3210
- sstream.tcc, 3210
- ssub\_match
  - Regular Expressions, 187
- stable\_partition
  - Mutating, 117
- stable\_sort
  - Sorting, 148
- stack, 3210
  - std::stack, 2707
- standard\_policies.hpp, 3211
- start
  - Numeric Arrays, 101
- state
  - std::fpos, 2241
- static\_pointer\_cast
  - std, 560
- std, 376
  - \_Construct, 507

`_Destroy`, 507, 508  
`__final_insertion_sort`, 498  
`__find`, 498  
`__find_if`, 498, 499  
`__find_if_not`, 499  
`__find_if_not_n`, 499  
`__gcd`, 499  
`__heap_select`, 499, 500  
`__inplace_stable_partition`, 500  
`__inplace_stable_sort`, 500  
`__insertion_sort`, 500  
`__introsort_loop`, 500, 501  
`__invoke`, 501  
`__iointit`, 565  
`__lg`, 501  
`__merge_adaptive`, 501  
`__merge_without_buffer`, 501, 502  
`__move_median_first`, 502  
`__move_merge`, 502  
`__move_merge_adaptive`, 502, 503  
`__move_merge_adaptive_backward`, 503  
`__partition`, 503  
`__reverse`, 503  
`__rotate`, 504  
`__rotate_adaptive`, 504  
`__search_n`, 504, 505  
`__stable_partition_adaptive`, 505  
`__umap_traits`, 496  
`__ummap_traits`, 496  
`__umset_traits`, 496  
`__unguarded_insertion_sort`, 505  
`__unguarded_linear_insert`, 505, 506  
`__unguarded_partition`, 506  
`__unguarded_partition_pivot`, 506  
`__unique_copy`, 506, 507  
`__uset_traits`, 496  
`accumulate`, 508  
`acos`, 508  
`acosh`, 508  
`adjacent_difference`, 509  
`advance`, 509  
`all`, 510  
`any`, 510  
`arg`, 510  
`asin`, 510  
`asinh`, 510  
`atan`, 511  
`atanh`, 511  
`begin`, 511  
`boolalpha`, 512  
`cerr`, 565  
`cin`, 565  
`clog`, 565  
`const_pointer_cast`, 512  
`count`, 512  
`cout`, 565  
`cref`, 512  
`dec`, 512  
`denorm_absent`, 497  
`denorm_indeterminate`, 497  
`denorm_present`, 497  
`distance`, 512  
`dynamic_pointer_cast`, 513  
`end`, 513, 514  
`endl`, 514  
`ends`, 514  
`fabs`, 514  
`fixed`, 514  
`flip`, 515  
`float_denorm_style`, 497  
`float_round_style`, 498  
`flush`, 515  
`get_money`, 515  
`get_temporary_buffer`, 515  
`getline`, 516, 517  
`hex`, 518  
`inner_product`, 518  
`internal`, 518  
`iota`, 519  
`isalnum`, 519  
`isalpha`, 519  
`isctrl`, 519  
`isdigit`, 519  
`isgraph`, 519  
`islower`, 519  
`isprint`, 519  
`ispunct`, 520  
`isspace`, 520  
`isupper`, 520  
`isxdigit`, 520  
`left`, 520  
`new_handler`, 496  
`noboolalpha`, 520  
`none`, 520  
`noshowbase`, 520  
`noshowpoint`, 521  
`noshowpos`, 521  
`noskipws`, 521  
`nounitbuf`, 521  
`nouppercase`, 521  
`oct`, 521  
`operator<`, 526d  
`operator<<`, 531d  
`operator<=<=`, 536  
`operator<=`, 536d  
`operator>`, 544, 545  
`operator>>`, 548d  
`operator>=>=`, 553

operator>=, [546](#), [547](#)  
operator~, [554](#)  
operator^, [554](#)  
operator+, [524d](#)  
operator==, [538d](#)  
operator&, [524](#)  
partial\_sum, [555](#)  
put\_money, [555](#)  
ref, [556](#)  
replace\_copy, [556](#)  
reset, [556](#)  
resetiosflags, [557](#)  
return\_temporary\_buffer, [557](#)  
right, [557](#)  
round\_to\_nearest, [498](#)  
round\_toward\_infinity, [498](#)  
round\_toward\_neg\_infinity, [498](#)  
round\_toward\_zero, [498](#)  
scientific, [557](#)  
set, [557](#)  
set\_new\_handler, [558](#)  
setbase, [558](#)  
setfill, [558](#)  
setiosflags, [558](#)  
setprecision, [558](#)  
setw, [559](#)  
showbase, [559](#)  
showpoint, [559](#)  
showpos, [559](#)  
size, [559](#)  
skipws, [560](#)  
static\_pointer\_cast, [560](#)  
streamoff, [496](#)  
streampos, [497](#)  
streamsize, [497](#)  
swap, [560](#), [561](#)  
test, [561](#)  
to\_string, [562](#)  
to\_ulong, [562](#)  
tolower, [562](#)  
toupper, [562](#)  
u16streampos, [497](#)  
u32streampos, [497](#)  
uninitialized\_copy, [563](#)  
uninitialized\_copy\_n, [563](#)  
uninitialized\_fill, [563](#)  
uninitialized\_fill\_n, [564](#)  
unitbuf, [564](#)  
uppercase, [564](#)  
wcerr, [565](#)  
wcin, [565](#)  
wclog, [565](#)  
wcout, [565](#)  
ws, [564](#)  
wstreampos, [497](#)  
std::\_\_atomic\_base<\_ITp >, [1283](#)  
std::\_\_atomic\_base<\_PTp \* >, [1285](#)  
std::\_\_atomic\_flag\_base, [1286](#)  
std::\_\_codecvt\_abstract\_base  
do\_out, [1288](#)  
in, [1289](#)  
out, [1289](#)  
unshift, [1290](#)  
std::\_\_ctype\_abstract\_base  
char\_type, [1293](#)  
do\_is, [1293](#)  
do\_narrow, [1294](#)  
do\_scan\_is, [1294](#)  
do\_scan\_not, [1295](#)  
do\_tolower, [1295](#), [1296](#)  
do\_toupper, [1296](#)  
do\_widen, [1297](#)  
is, [1298](#)  
narrow, [1298](#), [1299](#)  
scan\_is, [1299](#)  
scan\_not, [1300](#)  
tolower, [1300](#)  
toupper, [1301](#)  
widen, [1301](#), [1302](#)  
std::\_\_ctype\_abstract\_base<\_CharT >, [1291](#)  
std::\_\_debug, [565](#)  
operator<=, [570](#)  
operator>, [570](#)  
operator>=, [570](#)  
swap, [570](#)  
std::\_\_debug::bitset<\_Nb >, [1302](#)  
std::\_\_debug::deque  
\_M\_attach, [1307](#)  
\_M\_attach\_single, [1307](#)  
\_M\_const\_iterators, [1308](#)  
\_M\_detach, [1307](#)  
\_M\_detach\_all, [1307](#)  
\_M\_detach\_single, [1307](#)  
\_M\_detach\_singular, [1307](#)  
\_M\_get\_mutex, [1307](#)  
\_M\_invalidate\_all, [1307](#)  
\_M\_invalidate\_if, [1307](#)  
\_M\_iterators, [1308](#)  
\_M\_revalidate\_singular, [1308](#)  
\_M\_swap, [1308](#)  
\_M\_transfer\_from\_if, [1308](#)  
\_M\_version, [1308](#)  
std::\_\_debug::deque<\_Tp, \_Allocator >, [1304](#)  
std::\_\_debug::forward\_list  
\_M\_attach, [1311](#)  
\_M\_attach\_single, [1311](#)  
\_M\_const\_iterators, [1313](#)  
\_M\_detach, [1312](#)

- [\\_M\\_detach\\_all, 1312](#)
- [\\_M\\_detach\\_single, 1312](#)
- [\\_M\\_detach\\_singular, 1312](#)
- [\\_M\\_get\\_mutex, 1312](#)
- [\\_M\\_invalidate\\_all, 1312](#)
- [\\_M\\_invalidate\\_if, 1312](#)
- [\\_M\\_iterators, 1313](#)
- [\\_M\\_revalidate\\_singular, 1312](#)
- [\\_M\\_swap, 1312](#)
- [\\_M\\_transfer\\_from\\_if, 1312](#)
- [\\_M\\_version, 1313](#)
- [std::\\_\\_debug::forward\\_list< \\_Tp, \\_Alloc >, 1309](#)
- [std::\\_\\_debug::list](#)
  - [\\_M\\_attach, 1317](#)
  - [\\_M\\_attach\\_single, 1317](#)
  - [\\_M\\_const\\_iterators, 1318](#)
  - [\\_M\\_detach, 1317](#)
  - [\\_M\\_detach\\_all, 1317](#)
  - [\\_M\\_detach\\_single, 1317](#)
  - [\\_M\\_detach\\_singular, 1317](#)
  - [\\_M\\_get\\_mutex, 1317](#)
  - [\\_M\\_invalidate\\_all, 1317](#)
  - [\\_M\\_invalidate\\_if, 1317](#)
  - [\\_M\\_iterators, 1318](#)
  - [\\_M\\_revalidate\\_singular, 1317](#)
  - [\\_M\\_swap, 1317](#)
  - [\\_M\\_transfer\\_from\\_if, 1318](#)
  - [\\_M\\_version, 1318](#)
- [std::\\_\\_debug::list< \\_Tp, \\_Allocator >, 1314](#)
- [std::\\_\\_debug::map](#)
  - [\\_M\\_attach, 1321](#)
  - [\\_M\\_attach\\_single, 1321](#)
  - [\\_M\\_const\\_iterators, 1323](#)
  - [\\_M\\_detach, 1322](#)
  - [\\_M\\_detach\\_all, 1322](#)
  - [\\_M\\_detach\\_single, 1322](#)
  - [\\_M\\_detach\\_singular, 1322](#)
  - [\\_M\\_get\\_mutex, 1322](#)
  - [\\_M\\_invalidate\\_all, 1322](#)
  - [\\_M\\_invalidate\\_if, 1322](#)
  - [\\_M\\_iterators, 1323](#)
  - [\\_M\\_revalidate\\_singular, 1322](#)
  - [\\_M\\_swap, 1322](#)
  - [\\_M\\_transfer\\_from\\_if, 1322](#)
  - [\\_M\\_version, 1323](#)
- [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 1319](#)
- [std::\\_\\_debug::multimap](#)
  - [\\_M\\_attach, 1326](#)
  - [\\_M\\_attach\\_single, 1326](#)
  - [\\_M\\_const\\_iterators, 1328](#)
  - [\\_M\\_detach, 1327](#)
  - [\\_M\\_detach\\_all, 1327](#)
  - [\\_M\\_detach\\_single, 1327](#)
- [\\_M\\_detach\\_singular, 1327](#)
- [\\_M\\_get\\_mutex, 1327](#)
- [\\_M\\_invalidate\\_all, 1327](#)
- [\\_M\\_invalidate\\_if, 1327](#)
- [\\_M\\_iterators, 1328](#)
- [\\_M\\_revalidate\\_singular, 1327](#)
- [\\_M\\_swap, 1327](#)
- [\\_M\\_transfer\\_from\\_if, 1327](#)
- [\\_M\\_version, 1328](#)
- [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 1324](#)
- [std::\\_\\_debug::multiset](#)
  - [\\_M\\_attach, 1331](#)
  - [\\_M\\_attach\\_single, 1331](#)
  - [\\_M\\_const\\_iterators, 1333](#)
  - [\\_M\\_detach, 1331](#)
  - [\\_M\\_detach\\_all, 1332](#)
  - [\\_M\\_detach\\_single, 1332](#)
  - [\\_M\\_detach\\_singular, 1332](#)
  - [\\_M\\_get\\_mutex, 1332](#)
  - [\\_M\\_invalidate\\_all, 1332](#)
  - [\\_M\\_invalidate\\_if, 1332](#)
  - [\\_M\\_iterators, 1333](#)
  - [\\_M\\_revalidate\\_singular, 1332](#)
  - [\\_M\\_swap, 1332](#)
  - [\\_M\\_transfer\\_from\\_if, 1332](#)
  - [\\_M\\_version, 1333](#)
- [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >, 1329](#)
- [std::\\_\\_debug::set](#)
  - [\\_M\\_attach, 1336](#)
  - [\\_M\\_attach\\_single, 1336](#)
  - [\\_M\\_const\\_iterators, 1338](#)
  - [\\_M\\_detach, 1336](#)
  - [\\_M\\_detach\\_all, 1337](#)
  - [\\_M\\_detach\\_single, 1337](#)
  - [\\_M\\_detach\\_singular, 1337](#)
  - [\\_M\\_get\\_mutex, 1337](#)
  - [\\_M\\_invalidate\\_all, 1337](#)
  - [\\_M\\_invalidate\\_if, 1337](#)
  - [\\_M\\_iterators, 1338](#)
  - [\\_M\\_revalidate\\_singular, 1337](#)
  - [\\_M\\_swap, 1337](#)
  - [\\_M\\_transfer\\_from\\_if, 1337](#)
  - [\\_M\\_version, 1338](#)
- [std::\\_\\_debug::set< \\_Key, \\_Compare, \\_Allocator >, 1334](#)
- [std::\\_\\_debug::unordered\\_map](#)
  - [\\_M\\_attach, 1341](#)
  - [\\_M\\_attach\\_local, 1341](#)
  - [\\_M\\_attach\\_local\\_single, 1341](#)
  - [\\_M\\_attach\\_single, 1341](#)
  - [\\_M\\_const\\_iterators, 1342](#)
  - [\\_M\\_const\\_local\\_iterators, 1342](#)
  - [\\_M\\_detach, 1341](#)

- [\\_M\\_detach\\_all, 1341](#)
- [\\_M\\_detach\\_local, 1341](#)
- [\\_M\\_detach\\_local\\_single, 1341](#)
- [\\_M\\_detach\\_single, 1341](#)
- [\\_M\\_detach\\_singular, 1341](#)
- [\\_M\\_get\\_mutex, 1341](#)
- [\\_M\\_invalidate\\_all, 1341](#)
- [\\_M\\_invalidate\\_if, 1342](#)
- [\\_M\\_invalidate\\_local\\_if, 1342](#)
- [\\_M\\_iterators, 1342](#)
- [\\_M\\_local\\_iterators, 1343](#)
- [\\_M\\_revalidate\\_singular, 1342](#)
- [\\_M\\_swap, 1342](#)
- [\\_M\\_version, 1343](#)
- [std::\\_\\_debug::unordered\\_multimap](#)
  - [\\_M\\_attach, 1345](#)
  - [\\_M\\_attach\\_local, 1345](#)
  - [\\_M\\_attach\\_local\\_single, 1346](#)
  - [\\_M\\_attach\\_single, 1346](#)
  - [\\_M\\_const\\_iterators, 1347](#)
  - [\\_M\\_const\\_local\\_iterators, 1347](#)
  - [\\_M\\_detach, 1346](#)
  - [\\_M\\_detach\\_all, 1346](#)
  - [\\_M\\_detach\\_local, 1346](#)
  - [\\_M\\_detach\\_local\\_single, 1346](#)
  - [\\_M\\_detach\\_single, 1346](#)
  - [\\_M\\_detach\\_singular, 1346](#)
  - [\\_M\\_get\\_mutex, 1346](#)
  - [\\_M\\_invalidate\\_all, 1346](#)
  - [\\_M\\_invalidate\\_if, 1346](#)
  - [\\_M\\_invalidate\\_local\\_if, 1347](#)
  - [\\_M\\_iterators, 1347](#)
  - [\\_M\\_local\\_iterators, 1347](#)
  - [\\_M\\_revalidate\\_singular, 1347](#)
  - [\\_M\\_swap, 1347](#)
  - [\\_M\\_version, 1347](#)
- [std::\\_\\_debug::unordered\\_multiset](#)
  - [\\_M\\_attach, 1350](#)
  - [\\_M\\_attach\\_local, 1350](#)
  - [\\_M\\_attach\\_local\\_single, 1350](#)
  - [\\_M\\_attach\\_single, 1351](#)
  - [\\_M\\_const\\_iterators, 1352](#)
  - [\\_M\\_const\\_local\\_iterators, 1352](#)
  - [\\_M\\_detach, 1351](#)
  - [\\_M\\_detach\\_all, 1351](#)
  - [\\_M\\_detach\\_local, 1351](#)
  - [\\_M\\_detach\\_local\\_single, 1351](#)
  - [\\_M\\_detach\\_single, 1351](#)
  - [\\_M\\_detach\\_singular, 1351](#)
  - [\\_M\\_get\\_mutex, 1351](#)
  - [\\_M\\_invalidate\\_all, 1351](#)
  - [\\_M\\_invalidate\\_if, 1351](#)
  - [\\_M\\_invalidate\\_local\\_if, 1352](#)
  - [\\_M\\_iterators, 1352](#)
  - [\\_M\\_local\\_iterators, 1352](#)
  - [\\_M\\_revalidate\\_singular, 1352](#)
  - [\\_M\\_swap, 1352](#)
  - [\\_M\\_version, 1352](#)
- [std::\\_\\_debug::unordered\\_set](#)
  - [\\_M\\_attach, 1355](#)
  - [\\_M\\_attach\\_local, 1355](#)
  - [\\_M\\_attach\\_local\\_single, 1355](#)
  - [\\_M\\_attach\\_single, 1356](#)
  - [\\_M\\_const\\_iterators, 1357](#)
  - [\\_M\\_const\\_local\\_iterators, 1357](#)
  - [\\_M\\_detach, 1356](#)
  - [\\_M\\_detach\\_all, 1356](#)
  - [\\_M\\_detach\\_local, 1356](#)
  - [\\_M\\_detach\\_local\\_single, 1356](#)
  - [\\_M\\_detach\\_single, 1356](#)
  - [\\_M\\_detach\\_singular, 1356](#)
  - [\\_M\\_get\\_mutex, 1356](#)
  - [\\_M\\_invalidate\\_all, 1356](#)
  - [\\_M\\_invalidate\\_if, 1356](#)
  - [\\_M\\_invalidate\\_local\\_if, 1357](#)
  - [\\_M\\_iterators, 1357](#)
  - [\\_M\\_local\\_iterators, 1357](#)
  - [\\_M\\_revalidate\\_singular, 1357](#)
  - [\\_M\\_swap, 1357](#)
  - [\\_M\\_version, 1357](#)
- [std::\\_\\_debug::vector](#)
  - [\\_M\\_attach, 1361](#)
  - [\\_M\\_attach\\_single, 1361](#)
  - [\\_M\\_const\\_iterators, 1362](#)
  - [\\_M\\_detach, 1361](#)
  - [\\_M\\_detach\\_all, 1361](#)
  - [\\_M\\_detach\\_single, 1361](#)
  - [\\_M\\_detach\\_singular, 1361](#)
  - [\\_M\\_get\\_mutex, 1361](#)
  - [\\_M\\_invalidate\\_all, 1361](#)
  - [\\_M\\_invalidate\\_if, 1362](#)
  - [\\_M\\_iterators, 1362](#)
  - [\\_M\\_revalidate\\_singular, 1362](#)
  - [\\_M\\_swap, 1362](#)
  - [\\_M\\_transfer\\_from\\_if, 1362](#)
  - [\\_M\\_version, 1362](#)
- [vector, 1361](#)
- [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 1358](#)
- [std::\\_\\_detail, 570](#)
- [std::\\_\\_detail::\\_\\_Automaton, 1363](#)
- [std::\\_\\_detail::\\_\\_Before\\_begin< \\_NodeAlloc >, 1364](#)
- [std::\\_\\_detail::\\_\\_CharMatcher< \\_InIterT, \\_TraitsT >, 1364](#)
- [std::\\_\\_detail::\\_\\_Compiler< \\_InIter, \\_TraitsT >, 1365](#)
- [std::\\_\\_detail::\\_\\_Default\\_ranged\\_hash, 1366](#)
- [std::\\_\\_detail::\\_\\_EndTagger< \\_FwdIterT, \\_TraitsT >, 1366](#)
- [std::\\_\\_detail::\\_\\_Equality\\_base, 1369](#)
- [std::\\_\\_detail::\\_\\_Grep\\_matcher, 1369](#)
- [std::\\_\\_detail::\\_\\_Hash\\_node< \\_Value, false >, 1373](#)

- std::\_\_detail::\_\_Hash\_node< \_Value, true >, 1374
- std::\_\_detail::\_\_Hash\_node\_base, 1375
- std::\_\_detail::\_\_Hashtable\_ebo\_helper< \_Nm, \_Tp, false >, 1377
- std::\_\_detail::\_\_Hashtable\_ebo\_helper< \_Nm, \_Tp, true >, 1377
- std::\_\_detail::\_\_List\_node\_base, 1384
- std::\_\_detail::\_\_Mod\_range\_hashing, 1391
- std::\_\_detail::\_\_Nfa, 1392
  - \_M\_allocate\_and\_copy, 1395
  - \_M\_range\_check, 1395
  - assign, 1395, 1396
  - at, 1396
  - back, 1397
  - begin, 1397
  - capacity, 1397
  - cbegin, 1397
  - cend, 1397
  - clear, 1398
  - crbegin, 1398
  - crend, 1398
  - data, 1398
  - emplace, 1398
  - empty, 1399
  - end, 1399
  - erase, 1399
  - front, 1400
  - insert, 1400, 1401
  - max\_size, 1401
  - pop\_back, 1402
  - push\_back, 1402
  - rbegin, 1403
  - rend, 1403
  - reserve, 1403
  - resize, 1403, 1404
  - shrink\_to\_fit, 1404
  - size, 1404
  - swap, 1404
- std::\_\_detail::\_\_PatternCursor, 1408
- std::\_\_detail::\_\_Prime\_rehash\_policy, 1409
- std::\_\_detail::\_\_RangeMatcher< \_InIterT, \_TraitsT >, 1409
- std::\_\_detail::\_\_Results, 1411
- std::\_\_detail::\_\_Scanner
  - \_TokenT, 1413
- std::\_\_detail::\_\_Scanner< \_InputIterator >, 1412
- std::\_\_detail::\_\_Scanner\_base, 1414
- std::\_\_detail::\_\_SpecializedCursor< \_FwdIterT >, 1415
- std::\_\_detail::\_\_SpecializedResults< \_FwdIterT, \_Alloc >, 1416
- std::\_\_detail::\_\_StartTagger< \_FwdIterT, \_TraitsT >, 1416
- std::\_\_detail::\_\_State, 1417
- std::\_\_detail::\_\_StateSeq, 1418
- std::\_\_exception\_ptr::exception\_ptr, 1418
- std::\_\_has\_iterator\_category\_helper< \_Tp >, 1419
- std::\_\_is\_location\_invariant< \_Tp >, 1419
- std::\_\_numeric\_limits\_base, 1420
  - digits, 1421
  - digits10, 1421
  - has\_denorm, 1421
  - has\_denorm\_loss, 1421
  - has\_infinity, 1422
  - has\_quiet\_NaN, 1422
  - has\_signaling\_NaN, 1422
  - is\_bounded, 1422
  - is\_exact, 1422
  - is\_iec559, 1422
  - is\_integer, 1422
  - is\_modulo, 1422
  - is\_signed, 1422
  - is\_specialized, 1423
  - max\_digits10, 1423
  - max\_exponent, 1423
  - max\_exponent10, 1423
  - min\_exponent, 1423
  - min\_exponent10, 1423
  - radix, 1423
  - round\_style, 1423
  - tinyness\_before, 1423
  - traps, 1424
- std::\_\_parallel, 574
- std::\_\_parallel::\_\_CRandNumber< \_MustBeInt >, 1424
- std::\_\_profile, 591
  - operator<=, 596
  - operator>, 596
  - operator>=, 596
  - swap, 596
- std::\_\_profile::bitset< \_Nb >, 1424
- std::\_\_profile::deque< \_Tp, \_Allocator >, 1426
- std::\_\_profile::forward\_list< \_Tp, \_Alloc >, 1428
- std::\_\_profile::list< \_Tp, \_Allocator >, 1429
- std::\_\_profile::map< \_Key, \_Tp, \_Compare, \_Allocator >, 1431
- std::\_\_profile::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, 1433
- std::\_\_profile::multiset< \_Key, \_Compare, \_Allocator >, 1435
- std::\_\_profile::set< \_Key, \_Compare, \_Allocator >, 1436
- std::\_\_Base\_bitset
  - \_M\_w, 1444
- std::\_\_Base\_bitset< 0 >, 1445
- std::\_\_Base\_bitset< 1 >, 1446
- std::\_\_Base\_bitset< \_Nw >, 1443
- std::\_\_Deque\_base
  - \_M\_initialize\_map, 1448
- std::\_\_Deque\_base< \_Tp, \_Alloc >, 1447
- std::\_\_Deque\_iterator
  - \_M\_set\_node, 1450
- std::\_\_Deque\_iterator< \_Tp, \_Ref, \_Ptr >, 1449

- std::\_Derives\_from\_binary\_function< \_Tp >, 1451
- std::\_Derives\_from\_unary\_function< \_Tp >, 1451
- std::\_Function\_base, 1452
- std::\_Fwd\_list\_base< \_Tp, \_Alloc >, 1453
- std::\_Fwd\_list\_const\_iterator< \_Tp >, 1454
- std::\_Fwd\_list\_iterator< \_Tp >, 1455
- std::\_Fwd\_list\_node< \_Tp >, 1456
- std::\_Fwd\_list\_node\_base, 1457
- std::\_List\_base< \_Tp, \_Alloc >, 1462
- std::\_List\_const\_iterator< \_Tp >, 1463
- std::\_List\_iterator< \_Tp >, 1464
- std::\_List\_node
  - \_M\_data, 1466
- std::\_List\_node< \_Tp >, 1465
- std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_ArgTypes >, 1467
- std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1 >, 1468
- std::\_Maybe\_wrap\_member\_pointer< \_Tp >, 1470
- std::\_Maybe\_wrap\_member\_pointer< \_Tp \_Class::\* >, 1470
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_ArgTypes...) const >, 1471
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_ArgTypes...) const volatile >, 1472
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_ArgTypes...) volatile >, 1473
- std::\_Mem\_fn< \_Res(\_Class::\*)(\_ArgTypes...) >, 1474
- std::\_Mu< \_Arg, false, false >, 1475
- std::\_Mu< \_Arg, false, true >, 1475
- std::\_Mu< \_Arg, true, false >, 1476
- std::\_Mu< reference\_wrapper< \_Tp >, false, false >, 1476
- std::\_Placeholder< \_Num >, 1477
- std::\_Reference\_wrapper\_base< \_Tp >, 1477
- std::\_Safe\_tuple\_element< \_\_i, \_Tuple >, 1477
- std::\_Safe\_tuple\_element\_impl< \_\_i, \_Tuple, false >, 1479
- std::\_Temporary\_buffer
  - \_Temporary\_buffer, 1480
  - begin, 1480
  - end, 1480
  - requested\_size, 1481
  - size, 1481
- std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1479
- std::\_Tuple\_impl< \_Idx >, 1481
- std::\_Tuple\_impl< \_Idx, \_Head, \_Tail... >, 1482
- std::\_Vector\_base< \_Tp, \_Alloc >, 1484
- std::\_Weak\_result\_type< \_Functor >, 1485
- std::\_Weak\_result\_type\_impl< \_Functor >, 1486
- std::\_Weak\_result\_type\_impl< \_Res(\*)(\_ArgTypes...) >, 1486
- std::\_Weak\_result\_type\_impl< \_Res(&)(\_ArgTypes...) >, 1486
- std::\_Weak\_result\_type\_impl< \_Res(\_ArgTypes...) >, 1487
- std::add\_const< \_Tp >, 1488
- std::add\_cv< \_Tp >, 1489
- std::add\_lvalue\_reference< \_Tp >, 1489
- std::add\_pointer< \_Tp >, 1490
- std::add\_rvalue\_reference< \_Tp >, 1490
- std::add\_volatile< \_Tp >, 1490
- std::adopt\_lock\_t, 1491
- std::aligned\_storage< \_Len, \_Align >, 1491
- std::alignment\_of< \_Tp >, 1492
- std::allocator< \_Tp >, 1493
- std::allocator< void >, 1494
- std::allocator\_arg\_t, 1495
- std::allocator\_traits
  - allocate, 1498
  - allocator\_type, 1496
  - const\_pointer, 1496
  - const\_void\_pointer, 1496
  - construct, 1498
  - deallocate, 1498
  - destroy, 1499
  - difference\_type, 1496
  - max\_size, 1499
  - pointer, 1497
  - propagate\_on\_container\_copy\_assignment, 1497
  - propagate\_on\_container\_move\_assignment, 1497
  - propagate\_on\_container\_swap, 1497
  - select\_on\_container\_copy\_construction, 1499
  - size\_type, 1497
  - value\_type, 1497
  - void\_pointer, 1497
- std::allocator\_traits< \_Alloc >, 1495
- std::array< \_Tp, \_Nm >, 1500
- std::atomic< \_Tp >, 1501
- std::atomic< \_Tp \* >, 1502
- std::atomic< bool >, 1504
- std::atomic< char >, 1505
- std::atomic< char16\_t >, 1507
- std::atomic< char32\_t >, 1509
- std::atomic< int >, 1511
- std::atomic< long >, 1513
- std::atomic< long long >, 1515
- std::atomic< short >, 1517
- std::atomic< signed char >, 1519
- std::atomic< unsigned char >, 1521
- std::atomic< unsigned int >, 1523
- std::atomic< unsigned long >, 1525
- std::atomic< unsigned long long >, 1527
- std::atomic< unsigned short >, 1529
- std::atomic< wchar\_t >, 1531
- std::atomic\_bool, 1533
- std::atomic\_flag, 1534
- std::auto\_ptr

- [~auto\\_ptr, 1537](#)
  - [auto\\_ptr, 1536, 1537](#)
  - [element\\_type, 1536](#)
  - [get, 1537](#)
  - [operator\\*, 1537](#)
  - [operator->, 1538](#)
  - [operator=, 1538](#)
  - [release, 1538](#)
  - [reset, 1539](#)
- [std::auto\\_ptr<\\_Tp>, 1535](#)
- [std::auto\\_ptr\\_ref<\\_Tp1>, 1539](#)
- [std::back\\_insert\\_iterator](#)
  - [back\\_insert\\_iterator, 1541](#)
  - [container\\_type, 1541](#)
  - [difference\\_type, 1541](#)
  - [iterator\\_category, 1541](#)
  - [operator\\*, 1542](#)
  - [operator++, 1542](#)
  - [operator=, 1542](#)
  - [pointer, 1541](#)
  - [reference, 1541](#)
  - [value\\_type, 1541](#)
- [std::back\\_insert\\_iterator<\\_Container>, 1540](#)
- [std::bad\\_alloc, 1543](#)
  - [what, 1543](#)
- [std::bad\\_cast, 1544](#)
  - [what, 1544](#)
- [std::bad\\_exception, 1545](#)
  - [what, 1545](#)
- [std::bad\\_function\\_call, 1546](#)
  - [what, 1546](#)
- [std::bad\\_typeid, 1547](#)
  - [what, 1547](#)
- [std::bad\\_weak\\_ptr, 1548](#)
  - [what, 1548](#)
- [std::basic\\_filebuf](#)
  - [~basic\\_filebuf, 1552](#)
  - [\\_M\\_buf, 1566](#)
  - [\\_M\\_buf\\_locale, 1566](#)
  - [\\_M\\_buf\\_size, 1566](#)
  - [\\_M\\_create\\_pback, 1552](#)
  - [\\_M\\_destroy\\_pback, 1552](#)
  - [\\_M\\_ext\\_buf, 1567](#)
  - [\\_M\\_ext\\_buf\\_size, 1567](#)
  - [\\_M\\_ext\\_next, 1567](#)
  - [\\_M\\_in\\_beg, 1567](#)
  - [\\_M\\_in\\_cur, 1567](#)
  - [\\_M\\_in\\_end, 1567](#)
  - [\\_M\\_mode, 1567](#)
  - [\\_M\\_out\\_beg, 1568](#)
  - [\\_M\\_out\\_cur, 1568](#)
  - [\\_M\\_out\\_end, 1568](#)
  - [\\_M\\_pback, 1568](#)
  - [\\_M\\_pback\\_cur\\_save, 1568](#)
  - [\\_M\\_pback\\_end\\_save, 1569](#)
  - [\\_M\\_pback\\_init, 1569](#)
  - [\\_M\\_reading, 1569](#)
  - [\\_M\\_set\\_buffer, 1552](#)
- [basic\\_filebuf, 1552](#)
- [close, 1552](#)
- [eback, 1553](#)
- [egptr, 1553](#)
- [epptr, 1553](#)
- [gbump, 1554](#)
- [getloc, 1554](#)
- [gptr, 1554](#)
- [imbue, 1555](#)
- [in\\_avail, 1555](#)
- [is\\_open, 1555](#)
- [open, 1555, 1556](#)
- [overflow, 1556](#)
- [pbackfail, 1557](#)
- [pbase, 1557](#)
- [pbump, 1558](#)
- [pptr, 1558](#)
- [pubimbue, 1558](#)
- [pubseekoff, 1559](#)
- [pubseekpos, 1559](#)
- [pubsetbuf, 1559](#)
- [pubsync, 1559](#)
- [sbumpc, 1559](#)
- [seekoff, 1560](#)
- [seekpos, 1560](#)
- [setbuf, 1560](#)
- [setg, 1561](#)
- [setp, 1561](#)
- [sgetc, 1562](#)
- [sgetn, 1562](#)
- [showmanyc, 1562](#)
- [snextc, 1563](#)
- [sputbackc, 1563](#)
- [sputc, 1563](#)
- [sputn, 1564](#)
- [sungetc, 1564](#)
- [sync, 1564](#)
- [uflow, 1564](#)
- [underflow, 1565](#)
- [xsgetn, 1565](#)
- [xspn, 1566](#)

- [std::basic\\_filebuf<\\_CharT, \\_Traits>, 1549](#)
- [std::basic\\_fstream](#)
  - [~basic\\_fstream, 1579](#)
  - [\\_M\\_gcount, 1613](#)
  - [\\_M\\_getloc, 1579](#)
  - [\\_M\\_write, 1579](#)
  - [\\_\\_num\\_put\\_type, 1576](#)
- [adjustfield, 1613](#)
- [app, 1613](#)

ate, 1613  
bad, 1580  
badbit, 1613  
basefield, 1613  
basic\_fstream, 1578, 1579  
beg, 1614  
binary, 1614  
boolalpha, 1614  
clear, 1580  
close, 1580  
copyfmt, 1580  
cur, 1614  
dec, 1614  
end, 1614  
eof, 1581  
eofbit, 1614  
event, 1578  
event\_callback, 1576  
exceptions, 1581  
fail, 1582  
failbit, 1615  
fill, 1582  
fixed, 1615  
flags, 1583  
floatfield, 1615  
flush, 1583  
fmtflags, 1576  
gcount, 1583  
get, 1584d  
getline, 1586, 1587  
getloc, 1587  
good, 1587  
goodbit, 1615  
hex, 1615  
ignore, 1588  
imbue, 1589  
in, 1616  
init, 1589  
internal, 1616  
iostate, 1577  
is\_open, 1589  
iword, 1590  
left, 1616  
narrow, 1590  
oct, 1616  
open, 1590, 1591  
openmode, 1577  
operator void \*, 1591  
operator<<, 1591d  
operator>>, 1596d  
out, 1616  
peek, 1601  
precision, 1601, 1602  
put, 1602  
putback, 1602  
pword, 1603  
rdbuf, 1603, 1604  
rdstate, 1604  
read, 1604  
readsome, 1605  
register\_callback, 1605  
right, 1616  
scientific, 1617  
seekdir, 1578  
seekg, 1606  
seekp, 1606, 1607  
setf, 1607  
setstate, 1608  
showbase, 1617  
showpoint, 1617  
showpos, 1617  
skipws, 1617  
sync, 1608  
sync\_with\_stdio, 1609  
tellg, 1609  
tellp, 1609  
tie, 1610  
trunc, 1617  
unget, 1610  
unitbuf, 1617  
unsetf, 1611  
uppercase, 1617  
widen, 1611  
width, 1611  
write, 1612  
xalloc, 1612  
std::basic\_fstream<\_CharT, \_Traits >, 1570  
std::basic\_ifstream  
    ~basic\_ifstream, 1626  
    \_M\_gcount, 1653  
    \_M\_getloc, 1626  
    \_\_num\_put\_type, 1623  
adjustfield, 1653  
app, 1653  
ate, 1653  
bad, 1626  
badbit, 1653  
basefield, 1653  
basic\_ifstream, 1625  
beg, 1654  
binary, 1654  
boolalpha, 1654  
clear, 1626  
close, 1627  
copyfmt, 1627  
cur, 1654  
dec, 1654  
end, 1654

eof, 1627  
eofbit, 1654  
event, 1625  
event\_callback, 1623  
exceptions, 1628  
fail, 1628  
failbit, 1655  
fill, 1629  
fixed, 1655  
flags, 1629  
floatfield, 1655  
fmtflags, 1623  
gcount, 1630  
get, 1630d  
getline, 1632, 1633  
getloc, 1633  
good, 1634  
goodbit, 1655  
hex, 1655  
ignore, 1634, 1635  
imbue, 1635  
in, 1656  
init, 1636  
internal, 1656  
iostate, 1624  
is\_open, 1636  
iword, 1636  
left, 1656  
narrow, 1636  
oct, 1656  
open, 1637  
openmode, 1624  
operator void \*, 1637  
operator > >, 1638d  
out, 1656  
peek, 1643  
precision, 1643  
putback, 1644  
pword, 1644  
rdbuf, 1645  
rdstate, 1645  
read, 1645  
readsome, 1646  
register\_callback, 1647  
right, 1656  
scientific, 1657  
seekdir, 1624  
seekg, 1647  
setf, 1648  
setstate, 1648  
showbase, 1657  
showpoint, 1657  
showpos, 1657  
skipws, 1657  
sync, 1649  
sync\_with\_stdio, 1649  
tellg, 1650  
tie, 1650  
trunc, 1657  
unget, 1651  
unitbuf, 1657  
unsetf, 1651  
uppercase, 1657  
widen, 1651  
width, 1652  
xalloc, 1652  
std::basic\_ifstream< \_CharT, \_Traits >, 1618  
std::basic\_ios  
    ~basic\_ios, 1664  
    \_M\_getloc, 1665  
    \_\_ctype\_type, 1661  
    \_\_num\_get\_type, 1661  
    \_\_num\_put\_type, 1661  
    adjustfield, 1676  
    app, 1676  
    ate, 1676  
    bad, 1665  
    badbit, 1676  
    basefield, 1677  
    basic\_ios, 1664  
    beg, 1677  
    binary, 1677  
    boolalpha, 1677  
    char\_type, 1662  
    clear, 1665  
    copyfmt, 1665  
    cur, 1677  
    dec, 1677  
    end, 1678  
    eof, 1666  
    eofbit, 1678  
    event, 1664  
    event\_callback, 1662  
    exceptions, 1666  
    fail, 1667  
    failbit, 1678  
    fill, 1667  
    fixed, 1678  
    flags, 1668  
    floatfield, 1678  
    fmtflags, 1662  
    getloc, 1668  
    good, 1668  
    goodbit, 1679  
    hex, 1679  
    imbue, 1669  
    in, 1679  
    init, 1669

int\_type, 1663  
internal, 1679  
iostate, 1663  
iword, 1669  
left, 1679  
narrow, 1670  
oct, 1679  
off\_type, 1663  
openmode, 1663  
operator void \*, 1670  
out, 1680  
pos\_type, 1663  
precision, 1670, 1671  
pword, 1671  
rdbuf, 1671, 1672  
rdstate, 1672  
register\_callback, 1672  
right, 1680  
scientific, 1680  
seekdir, 1664  
setf, 1673  
setstate, 1673  
showbase, 1680  
showpoint, 1680  
showpos, 1680  
skipws, 1680  
sync\_with\_stdio, 1674  
tie, 1674  
traits\_type, 1664  
trunc, 1681  
unitbuf, 1681  
unsetf, 1675  
uppercase, 1681  
widen, 1675  
width, 1675, 1676  
xalloc, 1676  
std::basic\_ios< \_CharT, \_Traits >, 1658  
std::basic\_istream  
  ~basic\_istream, 1689  
  \_M\_gcount, 1722  
  \_M\_getloc, 1689  
  \_M\_write, 1690  
  \_\_num\_put\_type, 1687  
adjustfield, 1723  
app, 1723  
ate, 1723  
bad, 1690  
badbit, 1723  
basefield, 1723  
basic\_istream, 1689  
beg, 1723  
binary, 1724  
boolalpha, 1724  
clear, 1690  
copyfmt, 1690  
cur, 1724  
dec, 1724  
end, 1724  
eof, 1691  
eofbit, 1724  
event, 1689  
event\_callback, 1687  
exceptions, 1691  
fail, 1692  
failbit, 1725  
fill, 1692  
fixed, 1725  
flags, 1693  
floatfield, 1725  
flush, 1693  
fmtflags, 1687  
gcount, 1693  
get, 1694d  
getline, 1696, 1697  
getloc, 1697  
good, 1697  
goodbit, 1725  
hex, 1725  
ignore, 1698, 1699  
imbue, 1699  
in, 1725  
init, 1699  
internal, 1726  
iostate, 1688  
iword, 1700  
left, 1726  
narrow, 1700  
oct, 1726  
openmode, 1688  
operator void \*, 1700  
operator<<, 1701d  
operator>>, 1705d  
out, 1726  
peek, 1710  
precision, 1711  
put, 1711  
putback, 1712  
pword, 1712  
rdbuf, 1713  
rdstate, 1713  
read, 1714  
readsome, 1714  
register\_callback, 1715  
right, 1726  
scientific, 1726  
seekdir, 1688  
seekg, 1715  
seekp, 1716

- setf, 1717
- setstate, 1717
- showbase, 1727
- showpoint, 1727
- showpos, 1727
- skipws, 1727
- sync, 1718
- sync\_with\_stdio, 1718
- tellg, 1719
- tellp, 1719
- tie, 1719
- trunc, 1727
- unget, 1720
- unitbuf, 1727
- unsetf, 1720
- uppercase, 1727
- widen, 1720
- width, 1721
- write, 1722
- xalloc, 1722
- std::basic\_iostream< \_CharT, \_Traits >, 1681
- std::basic\_istream
  - ~basic\_istream, 1735
  - \_M\_gcount, 1761
  - \_M\_getloc, 1735
  - \_\_num\_put\_type, 1732
  - adjustfield, 1761
  - app, 1761
  - ate, 1761
  - bad, 1735
  - badbit, 1761
  - basefield, 1762
  - basic\_istream, 1735
  - beg, 1762
  - binary, 1762
  - boolalpha, 1762
  - clear, 1735
  - copyfmt, 1736
  - cur, 1762
  - dec, 1762
  - end, 1762
  - eof, 1736
  - eofbit, 1763
  - event, 1734
  - event\_callback, 1732
  - exceptions, 1736, 1737
  - fail, 1737
  - failbit, 1763
  - fill, 1737, 1738
  - fixed, 1763
  - flags, 1738
  - floatfield, 1763
  - fmtflags, 1733
  - gcount, 1739
  - get, 1739d
  - getline, 1741, 1742
  - getloc, 1742
  - good, 1743
  - goodbit, 1763
  - hex, 1764
  - ignore, 1743, 1744
  - imbue, 1744
  - in, 1764
  - init, 1745
  - internal, 1764
  - iostate, 1733
  - isword, 1745
  - left, 1764
  - narrow, 1745
  - oct, 1764
  - openmode, 1734
  - operator void \*, 1746
  - operator>>, 1746d
  - out, 1764
  - peek, 1751
  - precision, 1751
  - putback, 1752
  - pword, 1752
  - rdbuf, 1753
  - rdstate, 1753
  - read, 1754
  - readsome, 1754
  - register\_callback, 1755
  - right, 1765
  - scientific, 1765
  - seekdir, 1734
  - seekg, 1755
  - setf, 1756
  - setstate, 1757
  - showbase, 1765
  - showpoint, 1765
  - showpos, 1765
  - skipws, 1765
  - sync, 1757
  - sync\_with\_stdio, 1757
  - tellg, 1758
  - tie, 1758
  - trunc, 1765
  - unget, 1759
  - unitbuf, 1765
  - unsetf, 1759
  - uppercase, 1766
  - widen, 1759
  - width, 1760
  - xalloc, 1760
- std::basic\_istream< \_CharT, \_Traits >, 1728
- std::basic\_istream< \_CharT, \_Traits >::sentry, 1766
- std::basic\_istream::sentry

- operator bool, 1767
- sentry, 1767
- traits\_type, 1766
- std::basic\_istream
  - ~basic\_istream, 1776
  - \_M\_gcount, 1802
  - \_M\_getloc, 1776
  - \_\_num\_put\_type, 1773
  - adjustfield, 1802
  - app, 1802
  - ate, 1802
  - bad, 1776
  - badbit, 1802
  - basefield, 1802
  - basic\_istream, 1775
  - beg, 1803
  - binary, 1803
  - boolalpha, 1803
  - clear, 1776
  - copyfmt, 1777
  - cur, 1803
  - dec, 1803
  - end, 1803
  - eof, 1777
  - eofbit, 1803
  - event, 1775
  - event\_callback, 1773
  - exceptions, 1777
  - fail, 1778
  - failbit, 1804
  - fill, 1778
  - fixed, 1804
  - flags, 1779
  - floatfield, 1804
  - fmtflags, 1773
  - gcount, 1779
  - get, 1779d
  - getline, 1782, 1783
  - getloc, 1783
  - good, 1783
  - goodbit, 1804
  - hex, 1804
  - ignore, 1784
  - imbue, 1785
  - in, 1805
  - init, 1785
  - internal, 1805
  - iostate, 1774
  - isword, 1785
  - left, 1805
  - narrow, 1786
  - oct, 1805
  - openmode, 1774
  - operator void \*, 1786
  - operator >>, 1786d
  - out, 1805
  - peek, 1791
  - precision, 1792
  - putback, 1792
  - pword, 1793
  - rdbuf, 1793, 1794
  - rdstate, 1794
  - read, 1794
  - readsome, 1795
  - register\_callback, 1795
  - right, 1805
  - scientific, 1806
  - seekdir, 1774
  - seekg, 1795, 1796
  - setf, 1796, 1797
  - setstate, 1797
  - showbase, 1806
  - showpoint, 1806
  - showpos, 1806
  - skipws, 1806
  - str, 1797, 1798
  - sync, 1798
  - sync\_with\_stdio, 1798
  - tellg, 1799
  - tie, 1799
  - trunc, 1806
  - unget, 1800
  - unitbuf, 1806
  - unsetf, 1800
  - uppercase, 1806
  - widen, 1800
  - width, 1801
  - xalloc, 1801
- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1768
- std::basic\_ofstream
  - ~basic\_ofstream, 1814
  - \_M\_getloc, 1814
  - \_M\_write, 1815
  - \_\_num\_get\_type, 1811
  - adjustfield, 1835
  - app, 1835
  - ate, 1835
  - bad, 1815
  - badbit, 1835
  - basefield, 1835
  - basic\_ofstream, 1814
  - beg, 1835
  - binary, 1836
  - boolalpha, 1836
  - clear, 1815
  - close, 1816
  - copyfmt, 1816
  - cur, 1836

dec, 1836  
end, 1836  
eof, 1816  
eofbit, 1836  
event, 1813  
event\_callback, 1812  
exceptions, 1816, 1817  
fail, 1817  
failbit, 1837  
fill, 1817, 1818  
fixed, 1837  
flags, 1818  
floatfield, 1837  
flush, 1819  
fmtflags, 1812  
getloc, 1819  
good, 1819  
goodbit, 1837  
hex, 1837  
imbue, 1819  
in, 1838  
init, 1820  
internal, 1838  
iostate, 1812  
is\_open, 1820  
iword, 1820  
left, 1838  
narrow, 1821  
oct, 1838  
open, 1821  
openmode, 1813  
operator void \*, 1822  
operator<<, 1822d  
out, 1838  
precision, 1827  
put, 1828  
pword, 1828  
rdbuf, 1829  
rdstate, 1829  
register\_callback, 1829  
right, 1838  
scientific, 1838  
seekdir, 1813  
seekp, 1830  
setf, 1830, 1831  
setstate, 1831  
showbase, 1839  
showpoint, 1839  
showpos, 1839  
skipws, 1839  
sync\_with\_stdio, 1831  
tellp, 1832  
tie, 1832  
trunc, 1839  
unitbuf, 1839  
unsetf, 1833  
uppercase, 1839  
widen, 1833  
width, 1833, 1834  
write, 1834  
xalloc, 1834  
std::basic\_ofstream< \_CharT, \_Traits >, 1807  
std::basic\_ostream  
    ~basic\_ostream, 1847  
    \_M\_getloc, 1847  
    \_M\_write, 1847  
    \_\_num\_get\_type, 1844  
adjustfield, 1866  
app, 1866  
ate, 1866  
bad, 1847  
badbit, 1866  
basefield, 1866  
basic\_ostream, 1847  
beg, 1866  
binary, 1867  
boolalpha, 1867  
clear, 1848  
copyfmt, 1848  
cur, 1867  
dec, 1867  
end, 1867  
eof, 1848  
eofbit, 1867  
event, 1846  
event\_callback, 1844  
exceptions, 1848, 1849  
fail, 1849  
failbit, 1868  
fill, 1850  
fixed, 1868  
flags, 1850  
floatfield, 1868  
flush, 1851  
fmtflags, 1845  
getloc, 1851  
good, 1851  
goodbit, 1868  
hex, 1868  
imbue, 1851  
in, 1869  
init, 1852  
internal, 1869  
iostate, 1845  
iword, 1852  
left, 1869  
narrow, 1852  
oct, 1869

- openmode, 1846
- operator void \*, 1853
- operator<<, 1853d
- out, 1869
- precision, 1858
- put, 1859
- pword, 1859
- rdbuf, 1859, 1860
- rdstate, 1860
- register\_callback, 1860
- right, 1869
- scientific, 1869
- seekdir, 1846
- seekp, 1861
- setf, 1861, 1862
- setstate, 1862
- showbase, 1870
- showpoint, 1870
- showpos, 1870
- skipws, 1870
- sync\_with\_stdio, 1862
- tellp, 1863
- tie, 1863
- trunc, 1870
- unitbuf, 1870
- unsetf, 1864
- uppercase, 1870
- widen, 1864
- width, 1864, 1865
- write, 1865
- xalloc, 1865
- std::basic\_ostream< \_CharT, \_Traits >, 1840
- std::basic\_ostream< \_CharT, \_Traits >::sentry, 1871
- std::basic\_ostream::sentry
  - ~sentry, 1871
  - operator bool, 1871
  - sentry, 1871
- std::basic\_ostringstream
  - ~basic\_ostringstream, 1880
  - \_M\_getloc, 1880
  - \_M\_write, 1880
  - \_\_num\_get\_type, 1877
  - adjustfield, 1899
  - app, 1899
  - ate, 1899
  - bad, 1880
  - badbit, 1899
  - basefield, 1900
  - basic\_ostringstream, 1879
  - beg, 1900
  - binary, 1900
  - boolalpha, 1900
  - clear, 1881
  - copyfmt, 1881
  - cur, 1900
  - dec, 1900
  - end, 1900
  - eof, 1881
  - eofbit, 1901
  - event, 1879
  - event\_callback, 1877
  - exceptions, 1881, 1882
  - fail, 1882
  - failbit, 1901
  - fill, 1883
  - fixed, 1901
  - flags, 1883
  - floatfield, 1901
  - flush, 1884
  - fmtflags, 1877
  - getloc, 1884
  - good, 1884
  - goodbit, 1901
  - hex, 1902
  - imbue, 1884
  - in, 1902
  - init, 1885
  - internal, 1902
  - iostate, 1878
  - isword, 1885
  - left, 1902
  - narrow, 1885
  - oct, 1902
  - openmode, 1878
  - operator void \*, 1886
  - operator<<, 1886d
  - out, 1902
  - precision, 1891
  - put, 1892
  - pword, 1892
  - rdbuf, 1892, 1893
  - rdstate, 1893
  - register\_callback, 1893
  - right, 1903
  - scientific, 1903
  - seekdir, 1878
  - seekp, 1894
  - setf, 1894, 1895
  - setstate, 1895
  - showbase, 1903
  - showpoint, 1903
  - showpos, 1903
  - skipws, 1903
  - str, 1895, 1896
  - sync\_with\_stdio, 1896
  - tellp, 1896
  - tie, 1896, 1897
  - trunc, 1903

- unitbuf, [1904](#)
- unsetf, [1897](#)
- uppercase, [1904](#)
- widen, [1897](#)
- width, [1898](#)
- write, [1898](#)
- xalloc, [1899](#)
- std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc >, [1872](#)
- std::basic\_regex
  - ~basic\_regex, [1907](#)
  - assign, [1908d](#)
  - basic\_regex, [1905d](#)
  - flags, [1910](#)
  - getloc, [1910](#)
  - imbue, [1910](#)
  - mark\_count, [1910](#)
  - operator=, [1911](#)
  - swap, [1911](#)
- std::basic\_regex<\_Ch\_type, \_Rx\_traits >, [1904](#)
- std::basic\_streambuf
  - ~basic\_streambuf, [1916](#)
  - \_M\_buf\_locale, [1929](#)
  - \_M\_in\_beg, [1929](#)
  - \_M\_in\_cur, [1929](#)
  - \_M\_in\_end, [1929](#)
  - \_M\_out\_beg, [1930](#)
  - \_M\_out\_cur, [1930](#)
  - \_M\_out\_end, [1930](#)
  - \_\_streambuf\_type, [1915](#)
  - basic\_streambuf, [1916](#)
  - char\_type, [1915](#)
  - eback, [1916](#)
  - egptr, [1917](#)
  - eptr, [1917](#)
  - gbump, [1917](#)
  - getloc, [1918](#)
  - gptr, [1918](#)
  - imbue, [1918](#)
  - in\_avail, [1919](#)
  - int\_type, [1915](#)
  - off\_type, [1915](#)
  - overflow, [1919](#)
  - pbackfail, [1919](#)
  - pbase, [1920](#)
  - pbump, [1920](#)
  - pos\_type, [1916](#)
  - pptr, [1921](#)
  - pubimbue, [1921](#)
  - pubseekoff, [1921](#)
  - pubseekpos, [1921](#)
  - pubsetbuf, [1922](#)
  - pubsync, [1922](#)
  - sbumpc, [1922](#)
  - seekoff, [1922](#)
  - seekpos, [1923](#)
  - setbuf, [1923](#)
  - setg, [1923](#)
  - setp, [1924](#)
  - sgetc, [1924](#)
  - sgetn, [1924](#)
  - showmanyc, [1925](#)
  - snextc, [1925](#)
  - sputbackc, [1925](#)
  - sputc, [1926](#)
  - sputn, [1926](#)
  - sungetc, [1926](#)
  - sync, [1927](#)
  - traits\_type, [1916](#)
  - uflow, [1927](#)
  - underflow, [1927](#)
  - xsgetn, [1928](#)
  - xspn, [1928](#)
- std::basic\_streambuf<\_CharT, \_Traits >, [1912](#)
- std::basic\_string
  - ~basic\_string, [1937](#)
  - append, [1937d](#)
  - assign, [1940d](#)
  - at, [1943](#)
  - back, [1943](#)
  - basic\_string, [1935d](#)
  - begin, [1944](#)
  - c\_str, [1944](#)
  - capacity, [1944](#)
  - cbegin, [1944](#)
  - cend, [1944](#)
  - clear, [1944](#)
  - compare, [1945d](#)
  - copy, [1947](#)
  - crbegin, [1948](#)
  - crend, [1948](#)
  - data, [1948](#)
  - empty, [1948](#)
  - end, [1948](#)
  - erase, [1949](#)
  - find, [1950](#), [1951](#)
  - find\_first\_not\_of, [1951](#), [1952](#)
  - find\_first\_of, [1953](#), [1954](#)
  - find\_last\_not\_of, [1954](#), [1955](#)
  - find\_last\_of, [1956](#), [1957](#)
  - front, [1957](#)
  - get\_allocator, [1957](#)
  - insert, [1957d](#)
  - length, [1961](#)
  - max\_size, [1962](#)
  - npos, [1974](#)
  - operator+=, [1962](#)
  - operator=, [1963](#), [1964](#)
  - pop\_back, [1964](#)

- push\_back, 1965
- rbegin, 1965
- rend, 1965
- replace, 1965d
- reserve, 1971
- resize, 1971, 1972
- rfind, 1972, 1973
- shrink\_to\_fit, 1973
- size, 1973
- substr, 1974
- swap, 1974
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1930
- std::basic\_stringbuf
  - \_M\_buf\_locale, 1990
  - \_M\_in\_beg, 1991
  - \_M\_in\_cur, 1991
  - \_M\_in\_end, 1991
  - \_M\_mode, 1991
  - \_M\_out\_beg, 1991
  - \_M\_out\_cur, 1991
  - \_M\_out\_end, 1991
  - basic\_stringbuf, 1977
  - eback, 1978
  - egptr, 1978
  - eptr, 1978
  - gbump, 1979
  - getloc, 1979
  - gptr, 1979
  - imbue, 1979
  - in\_avail, 1980
  - overflow, 1980
  - pbackfail, 1981
  - pbase, 1981
  - pbump, 1981
  - pptr, 1982
  - pubimbue, 1982
  - pubseekoff, 1982
  - pubseekpos, 1983
  - pubsetbuf, 1983
  - pubsync, 1983
  - sbumpc, 1983
  - seekoff, 1983
  - seekpos, 1984
  - setbuf, 1984
  - setg, 1984
  - setp, 1985
  - sgetc, 1985
  - sgetn, 1985
  - showmanyc, 1986
  - snextc, 1986
  - sputbackc, 1986
  - sputc, 1987
  - sputn, 1987
  - str, 1987, 1988
  - sungetc, 1988
  - sync, 1988
  - uflow, 1989
  - underflow, 1989
  - xsggetn, 1989
  - xspn, 1990
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1975
- std::basic\_stringstream
  - ~basic\_stringstream, 2001
  - \_M\_gcount, 2035
  - \_M\_getloc, 2001
  - \_M\_write, 2002
  - \_\_num\_put\_type, 1998
  - adjustfield, 2035
  - app, 2035
  - ate, 2035
  - bad, 2002
  - badbit, 2035
  - basefield, 2035
  - basic\_stringstream, 2001
  - beg, 2036
  - binary, 2036
  - boolalpha, 2036
  - clear, 2002
  - copyfmt, 2002
  - cur, 2036
  - dec, 2036
  - end, 2036
  - eof, 2003
  - eofbit, 2036
  - event, 2000
  - event\_callback, 1998
  - exceptions, 2003
  - fail, 2004
  - failbit, 2037
  - fill, 2004
  - fixed, 2037
  - flags, 2005
  - floatfield, 2037
  - flush, 2005
  - fmtflags, 1999
  - gcount, 2005
  - get, 2006d
  - getline, 2008, 2009
  - getloc, 2009
  - good, 2009
  - goodbit, 2037
  - hex, 2037
  - ignore, 2010, 2011
  - imbue, 2011
  - in, 2038
  - init, 2011
  - internal, 2038
  - iostate, 1999

- isword, 2012
  - left, 2038
  - narrow, 2012
  - oct, 2038
  - openmode, 2000
  - operator void \*, 2012
  - operator<<, 2013d
  - operator>>, 2017d
  - out, 2038
  - peek, 2022
  - precision, 2023
  - put, 2023
  - putback, 2024
  - pwd, 2024
  - rdbuf, 2025
  - rdstate, 2025
  - read, 2026
  - readsome, 2026
  - register\_callback, 2027
  - right, 2038
  - scientific, 2039
  - seekdir, 2000
  - seekg, 2027
  - seekp, 2028
  - setf, 2029
  - setstate, 2029
  - showbase, 2039
  - showpoint, 2039
  - showpos, 2039
  - skipws, 2039
  - str, 2030
  - sync, 2030
  - sync\_with\_stdio, 2031
  - tellg, 2031
  - tellp, 2031
  - tie, 2031, 2032
  - trunc, 2039
  - unget, 2032
  - unitbuf, 2039
  - unsetf, 2032
  - uppercase, 2039
  - widen, 2033
  - width, 2033
  - write, 2034
  - xalloc, 2034
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1992
- std::bernoulli\_distribution, 2040
  - bernoulli\_distribution, 2041
  - max, 2041
  - min, 2041
  - operator(), 2041
  - operator==, 2042
  - p, 2041
  - param, 2041, 2042
  - reset, 2042
  - result\_type, 2041
- std::bernoulli\_distribution::param\_type, 2042
- std::bidirectional\_iterator\_tag, 2043
- std::binary\_function
  - first\_argument\_type, 2045
  - result\_type, 2045
  - second\_argument\_type, 2045
- std::binary\_function< \_Arg1, \_Arg2, \_Result >, 2044
- std::binary\_negate
  - first\_argument\_type, 2046
  - result\_type, 2046
  - second\_argument\_type, 2046
- std::binary\_negate< \_Predicate >, 2045
- std::binder1st
  - argument\_type, 2048
  - result\_type, 2048
- std::binder1st< \_Operation >, 2047
- std::binder2nd
  - argument\_type, 2049
  - result\_type, 2049
- std::binder2nd< \_Operation >, 2048
- std::binomial\_distribution
  - max, 2051
  - min, 2051
  - operator<<, 2052
  - operator>>, 2052
  - operator(), 2051
  - operator==, 2052
  - p, 2051
  - param, 2051
  - reset, 2052
  - result\_type, 2051
  - t, 2052
- std::binomial\_distribution< \_IntType >, 2049
- std::binomial\_distribution< \_IntType >::param\_type, 2053
- std::cauchy\_distribution
  - max, 2055
  - min, 2055
  - operator(), 2055
  - operator==, 2055
  - param, 2055
  - reset, 2055
  - result\_type, 2054
- std::cauchy\_distribution< \_RealType >, 2053
- std::cauchy\_distribution< \_RealType >::param\_type, 2056
- std::char\_traits< \_\_gnu\_cxx::character< V, I, S > >, 2058
- std::char\_traits< \_CharT >, 2057
- std::char\_traits< char >, 2059
- std::char\_traits< wchar\_t >, 2060
- std::chi\_squared\_distribution
  - max, 2062
  - min, 2062

operator<<, 2063  
 operator>>, 2063  
 operator(), 2062  
 operator==, 2063  
 param, 2062  
 reset, 2062  
 result\_type, 2061  
 std::chi\_squared\_distribution< \_RealType >, 2060  
 std::chi\_squared\_distribution< \_RealType >::param\_type, 2063  
 std::chrono, 597  
     duration\_cast, 600  
     hours, 599  
     microseconds, 599  
     milliseconds, 599  
     minutes, 600  
     nanoseconds, 600  
     seconds, 600  
     time\_point\_cast, 600  
 std::chrono::duration< \_Rep, \_Period >, 2064  
 std::chrono::duration\_values< \_Rep >, 2065  
 std::chrono::system\_clock, 2066  
 std::chrono::time\_point< \_Clock, \_Dur >, 2066  
 std::chrono::treat\_as\_floating\_point< \_Rep >, 2067  
 std::codecvt  
     do\_out, 2069  
     in, 2070  
     out, 2070  
     unshift, 2071  
 std::codecvt< \_InternT, \_ExternT, \_StateT >, 2068  
 std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2072  
     do\_out, 2073  
     in, 2073  
     out, 2074  
     unshift, 2075  
 std::codecvt< char, char, mbstate\_t >, 2075  
     do\_out, 2077  
     in, 2077  
     out, 2078  
     unshift, 2078  
 std::codecvt< wchar\_t, char, mbstate\_t >, 2079  
     do\_out, 2080  
     in, 2081  
     out, 2081  
     unshift, 2082  
 std::codecvt\_base, 2083  
 std::codecvt\_byname  
     do\_out, 2085  
     in, 2086  
     out, 2086  
     unshift, 2087  
 std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2084  
 std::collate  
     ~collate, 2090  
     char\_type, 2089  
     collate, 2089, 2090  
     compare, 2090  
     do\_compare, 2090  
     do\_hash, 2091  
     do\_transform, 2091  
     hash, 2092  
     id, 2092  
     string\_type, 2089  
     transform, 2092  
 std::collate< \_CharT >, 2088  
 std::collate\_byname  
     char\_type, 2094  
     compare, 2094  
     do\_compare, 2095  
     do\_hash, 2095  
     do\_transform, 2096  
     hash, 2096  
     id, 2097  
     string\_type, 2094  
     transform, 2096  
 std::collate\_byname< \_CharT >, 2093  
 std::complex  
     complex, 2098  
     operator+=, 2098  
     operator-=, 2098  
     value\_type, 2098  
 std::complex< \_Tp >, 2097  
 std::complex< double >, 2099  
 std::complex< float >, 2099  
 std::complex< long double >, 2100  
 std::condition\_variable, 2101  
 std::condition\_variable\_any, 2102  
 std::conditional< \_Cond, \_Iftrue, \_Iffalse >, 2102  
 std::const\_mem\_fun1\_ref\_t  
     first\_argument\_type, 2103  
     result\_type, 2103  
     second\_argument\_type, 2104  
 std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2103  
 std::const\_mem\_fun1\_t  
     first\_argument\_type, 2105  
     result\_type, 2105  
     second\_argument\_type, 2105  
 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2104  
 std::const\_mem\_fun\_ref\_t  
     argument\_type, 2106  
     result\_type, 2106  
 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 2106  
 std::const\_mem\_fun\_t  
     argument\_type, 2108  
     result\_type, 2108  
 std::const\_mem\_fun\_t< \_Ret, \_Tp >, 2107  
 std::ctype

- do\_is, 2110
- do\_narrow, 2111
- do\_scan\_is, 2112
- do\_scan\_not, 2112
- do\_tolower, 2113
- do\_toupper, 2113, 2114
- do\_widen, 2114, 2115
- id, 2119
- is, 2115
- narrow, 2116
- scan\_is, 2116
- scan\_not, 2117
- tolower, 2117
- toupper, 2118
- widen, 2118, 2119
- std::ctype< \_CharT >, 2108
- std::ctype< char >, 2120
  - ~ctype, 2122
  - char\_type, 2122
  - classic\_table, 2122
  - ctype, 2122
  - do\_narrow, 2123
  - do\_tolower, 2123, 2124
  - do\_toupper, 2124
  - do\_widen, 2125
  - id, 2130
  - is, 2126
  - narrow, 2126, 2127
  - scan\_is, 2127
  - scan\_not, 2127
  - table, 2128
  - table\_size, 2130
  - tolower, 2128
  - toupper, 2128, 2129
  - widen, 2129, 2130
- std::ctype< wchar\_t >, 2131
  - ~ctype, 2133
  - char\_type, 2133
  - ctype, 2133
  - do\_is, 2134
  - do\_narrow, 2134, 2135
  - do\_scan\_is, 2135
  - do\_scan\_not, 2136
  - do\_tolower, 2136
  - do\_toupper, 2137
  - do\_widen, 2137, 2138
  - id, 2143
  - is, 2138, 2139
  - narrow, 2139
  - scan\_is, 2140
  - scan\_not, 2140
  - tolower, 2140, 2141
  - toupper, 2141, 2142
  - widen, 2142
- std::ctype\_base, 2143
- std::ctype\_byname
  - do\_is, 2146
  - do\_narrow, 2147
  - do\_scan\_is, 2147
  - do\_scan\_not, 2148
  - do\_tolower, 2148, 2149
  - do\_toupper, 2149
  - do\_widen, 2150
  - id, 2155
  - is, 2151
  - narrow, 2151, 2152
  - scan\_is, 2152
  - scan\_not, 2152
  - tolower, 2153
  - toupper, 2153, 2154
  - widen, 2154, 2155
- std::ctype\_byname< \_CharT >, 2144
- std::ctype\_byname< char >, 2156
  - char\_type, 2158
  - classic\_table, 2158
  - do\_narrow, 2158
  - do\_tolower, 2159
  - do\_toupper, 2159, 2160
  - do\_widen, 2160
  - id, 2166
  - is, 2161
  - narrow, 2162
  - scan\_is, 2162
  - scan\_not, 2163
  - table, 2163
  - table\_size, 2166
  - tolower, 2163, 2164
  - toupper, 2164
  - widen, 2165
- std::decay< \_Tp >, 2166
- std::decimal, 600
  - decimal32\_to\_long\_long, 609
- std::decimal::decimal128, 2166
  - decimal128, 2168
- std::decimal::decimal32, 2168
  - decimal32, 2169
- std::decimal::decimal64, 2169
  - decimal64, 2171
- std::default\_delete< \_Tp >, 2171
- std::defer\_lock\_t, 2172
- std::deque
  - ~deque, 2179
  - \_M\_fill\_initialize, 2179
  - \_M\_initialize\_map, 2180
  - \_M\_new\_elements\_at\_back, 2180
  - \_M\_new\_elements\_at\_front, 2180
  - \_M\_pop\_back\_aux, 2180
  - \_M\_pop\_front\_aux, 2181

- [\\_M\\_push\\_back\\_aux](#), 2181
- [\\_M\\_push\\_front\\_aux](#), 2181
- [\\_M\\_range\\_check](#), 2181
- [\\_M\\_range\\_initialize](#), 2181
- [\\_M\\_reallocate\\_map](#), 2182
- [\\_M\\_reserve\\_elements\\_at\\_back](#), 2182
- [\\_M\\_reserve\\_elements\\_at\\_front](#), 2182
- [\\_M\\_reserve\\_map\\_at\\_back](#), 2182
- [\\_M\\_reserve\\_map\\_at\\_front](#), 2183
- [assign](#), 2183
- [at](#), 2184
- [back](#), 2184, 2185
- [begin](#), 2185
- [cbegin](#), 2185
- [cend](#), 2185
- [clear](#), 2185
- [crbegin](#), 2186
- [crend](#), 2186
- [deque](#), 2177d
- [emplace](#), 2186
- [empty](#), 2186
- [end](#), 2186
- [erase](#), 2187
- [front](#), 2187
- [get\\_allocator](#), 2188
- [insert](#), 2188, 2189
- [max\\_size](#), 2189
- [operator=](#), 2189, 2190
- [pop\\_back](#), 2191
- [pop\\_front](#), 2191
- [push\\_back](#), 2191
- [push\\_front](#), 2191
- [rbegin](#), 2192
- [rend](#), 2192
- [resize](#), 2192, 2193
- [shrink\\_to\\_fit](#), 2193
- [size](#), 2193
- [swap](#), 2193
- [std::deque< \\_Tp, \\_Alloc >](#), 2172
- [std::discard\\_block\\_engine](#)
  - [base](#), 2196
  - [discard](#), 2196
  - [discard\\_block\\_engine](#), 2195, 2196
  - [max](#), 2196
  - [min](#), 2196
  - [operator<<](#), 2197
  - [operator>>](#), 2198
  - [operator\(\)](#), 2197
  - [operator==](#), 2198
  - [result\\_type](#), 2195
  - [seed](#), 2197
- [std::discard\\_block\\_engine< \\_RandomNumberEngine, \\_\\_-  
p, \\_\\_r >](#), 2194
- [std::discrete\\_distribution](#)
  - [max](#), 2200
  - [min](#), 2200
  - [operator<<](#), 2201
  - [operator>>](#), 2201
  - [operator\(\)](#), 2200
  - [operator==](#), 2201
  - [param](#), 2200
  - [probabilities](#), 2200
  - [reset](#), 2201
  - [result\\_type](#), 2200
- [std::discrete\\_distribution< \\_IntType >](#), 2198
- [std::discrete\\_distribution< \\_IntType >::param\\_type](#), 2202
- [std::divides](#)
  - [first\\_argument\\_type](#), 2203
  - [result\\_type](#), 2203
  - [second\\_argument\\_type](#), 2204
- [std::divides< \\_Tp >](#), 2203
- [std::domain\\_error](#), 2204
  - [what](#), 2204
- [std::enable\\_if< bool, \\_Tp >](#), 2205
- [std::enable\\_shared\\_from\\_this< \\_Tp >](#), 2205
- [std::equal\\_to](#)
  - [first\\_argument\\_type](#), 2207
  - [result\\_type](#), 2207
  - [second\\_argument\\_type](#), 2207
- [std::equal\\_to< \\_Tp >](#), 2206
- [std::error\\_category](#), 2207
- [std::error\\_code](#), 2208
- [std::error\\_condition](#), 2208
- [std::exception](#), 2210
  - [what](#), 2211
- [std::exponential\\_distribution](#)
  - [exponential\\_distribution](#), 2212
  - [lambda](#), 2212
  - [max](#), 2212
  - [min](#), 2212
  - [operator\(\)](#), 2213
  - [operator==](#), 2213
  - [param](#), 2213
  - [reset](#), 2213
  - [result\\_type](#), 2212
- [std::exponential\\_distribution< \\_RealType >](#), 2211
- [std::exponential\\_distribution< \\_RealType >::param\\_type](#), 2214
- [std::extent< typename, \\_Uint >](#), 2215
- [std::extreme\\_value\\_distribution](#)
  - [a](#), 2217
  - [b](#), 2217
  - [max](#), 2217
  - [min](#), 2217
  - [operator\(\)](#), 2217
  - [operator==](#), 2218
  - [param](#), 2217, 2218
  - [reset](#), 2218

result\_type, 2217  
 std::extreme\_value\_distribution< \_RealType >, 2216  
 std::extreme\_value\_distribution< \_RealType >::param\_type, 2218  
 std::fisher\_f\_distribution  
   max, 2220  
   min, 2220  
   operator<=, 2221  
   operator>=, 2221  
   operator(), 2220  
   operator==, 2221  
   param, 2220, 2221  
   reset, 2221  
   result\_type, 2220  
 std::fisher\_f\_distribution< \_RealType >, 2219  
 std::fisher\_f\_distribution< \_RealType >::param\_type, 2222  
 std::forward\_iterator\_tag, 2223  
 std::forward\_list  
   ~forward\_list, 2229  
   assign, 2229  
   before\_begin, 2230  
   begin, 2230  
   cbefore\_begin, 2230  
   cbegin, 2230  
   cend, 2231  
   clear, 2231  
   emplace\_after, 2231  
   emplace\_front, 2231  
   empty, 2232  
   end, 2232  
   erase\_after, 2232  
   forward\_list, 2227, 2228  
   front, 2233  
   get\_allocator, 2233  
   insert\_after, 2233, 2234  
   max\_size, 2235  
   merge, 2235  
   operator=, 2236  
   pop\_front, 2236  
   push\_front, 2236  
   remove, 2237  
   remove\_if, 2237  
   resize, 2237, 2238  
   reverse, 2238  
   sort, 2238  
   splice\_after, 2238, 2239  
   swap, 2239  
   unique, 2239, 2240  
 std::forward\_list< \_Tp, \_Alloc >, 2224  
 std::fpos  
   fpos, 2241  
   operator streamoff, 2241  
   operator+, 2241  
   operator+=, 2241  
   operator-, 2241  
   operator=, 2241  
   state, 2241  
 std::fpos< \_StateT >, 2240  
 std::front\_insert\_iterator  
   container\_type, 2243  
   difference\_type, 2243  
   front\_insert\_iterator, 2244  
   iterator\_category, 2243  
   operator\*, 2244  
   operator++, 2244  
   operator=, 2244  
   pointer, 2243  
   reference, 2243  
   value\_type, 2243  
 std::front\_insert\_iterator< \_Container >, 2242  
 std::function< \_Res(\_ArgTypes...)>, 2245  
   function, 2246, 2247  
   operator bool, 2248  
   operator(), 2248  
   operator=, 2248, 2249  
   swap, 2250  
   target, 2250  
   target\_type, 2250  
 std::future\_error, 2251  
   what, 2251  
 std::gamma\_distribution  
   alpha, 2253  
   beta, 2253  
   gamma\_distribution, 2253  
   max, 2254  
   min, 2254  
   operator<=, 2255  
   operator>=, 2255  
   operator(), 2254  
   operator==, 2255  
   param, 2254  
   reset, 2254  
   result\_type, 2253  
 std::gamma\_distribution< \_RealType >, 2252  
 std::gamma\_distribution< \_RealType >::param\_type, 2256  
 std::geometric\_distribution  
   max, 2257  
   min, 2257  
   operator(), 2257  
   operator==, 2258  
   p, 2258  
   param, 2258  
   reset, 2258  
   result\_type, 2257  
 std::geometric\_distribution< \_IntType >, 2256

- std::geometric\_distribution< \_IntType >::param\_type, 2259
- std::greater
  - first\_argument\_type, 2260
  - result\_type, 2260
  - second\_argument\_type, 2261
- std::greater< \_Tp >, 2260
- std::greater\_equal
  - first\_argument\_type, 2262
  - result\_type, 2262
  - second\_argument\_type, 2262
- std::greater\_equal< \_Tp >, 2261
- std::gslice, 2262
- std::gslice\_array< \_Tp >, 2263
- std::has\_trivial\_copy\_assign< \_Tp >, 2265
- std::has\_trivial\_copy\_constructor< \_Tp >, 2266
- std::has\_trivial\_default\_constructor< \_Tp >, 2267
- std::has\_virtual\_destructor< \_Tp >, 2268
- std::hash< \_\_debug::bitset< \_Nb > >, 2269
- std::hash< \_\_debug::vector< bool, \_Alloc > >, 2269
- std::hash< \_\_gnu\_cxx::\_\_u16vstring >, 2270
- std::hash< \_\_gnu\_cxx::\_\_u32vstring >, 2270
- std::hash< \_\_gnu\_cxx::\_\_vstring >, 2271
- std::hash< \_\_gnu\_cxx::\_\_wvstring >, 2271
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2272
  - argument\_type, 2273
  - result\_type, 2273
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2274
  - argument\_type, 2274
  - result\_type, 2275
- std::hash< \_\_profile::bitset< \_Nb > >, 2275
- std::hash< \_\_profile::vector< bool, \_Alloc > >, 2275
- std::hash< \_\_shared\_ptr< \_Tp, \_Lp > >, 2276
- std::hash< \_Tp \* >, 2276
- std::hash< bool >, 2277
- std::hash< char >, 2277
- std::hash< char16\_t >, 2278
- std::hash< char32\_t >, 2278
- std::hash< double >, 2279
- std::hash< error\_code >, 2279
- std::hash< float >, 2280
- std::hash< int >, 2280
- std::hash< long >, 2281
- std::hash< long double >, 2281
- std::hash< long long >, 2282
- std::hash< shared\_ptr< \_Tp > >, 2282
- std::hash< short >, 2283
- std::hash< signed char >, 2283
- std::hash< string >, 2284
- std::hash< thread::id >, 2284
- std::hash< type\_index >, 2285
- std::hash< u16string >, 2285
- std::hash< u32string >, 2286
- std::hash< unique\_ptr< \_Tp, \_Dp > >, 2286
- std::hash< unsigned char >, 2287
- std::hash< unsigned int >, 2287
- std::hash< unsigned long >, 2288
- std::hash< unsigned long long >, 2288
- std::hash< unsigned short >, 2289
- std::hash< wchar\_t >, 2289
- std::hash< wstring >, 2290
- std::hash<::bitset< \_Nb > >, 2290
- std::hash<::vector< bool, \_Alloc > >, 2291
- std::independent\_bits\_engine
  - base, 2293
  - discard, 2294
  - independent\_bits\_engine, 2292, 2293
  - max, 2294
  - min, 2294
  - operator>>, 2295
  - operator(), 2294
  - operator==, 2295
  - result\_type, 2292
  - seed, 2294
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2291
- std::indirect\_array< \_Tp >, 2295
- std::initializer\_list< \_E >, 2297
- std::input\_iterator\_tag, 2298
- std::insert\_iterator
  - container\_type, 2300
  - difference\_type, 2300
  - insert\_iterator, 2300
  - iterator\_category, 2300
  - operator\*, 2301
  - operator++, 2301
  - operator=, 2301
  - pointer, 2300
  - reference, 2300
  - value\_type, 2300
- std::insert\_iterator< \_Container >, 2299
- std::integral\_constant< \_Tp, \_\_v >, 2302
- std::invalid\_argument, 2303
  - what, 2304
- std::ios\_base, 2304
  - ~ios\_base, 2309
  - \_M\_getloc, 2309
  - adjustfield, 2313
  - app, 2314
  - ate, 2314
  - badbit, 2314
  - basefield, 2314
  - beg, 2314
  - binary, 2314
  - boolalpha, 2315
  - cur, 2315
  - dec, 2315
  - end, 2315

- eofbit, [2315](#)
- event, [2308](#)
- event\_callback, [2307](#)
- failbit, [2315](#)
- fixed, [2316](#)
- flags, [2309](#)
- floatfield, [2316](#)
- fmtflags, [2307](#)
- getloc, [2309](#)
- goodbit, [2316](#)
- hex, [2316](#)
- imbue, [2310](#)
- in, [2316](#)
- internal, [2317](#)
- iostate, [2307](#)
- isword, [2310](#)
- left, [2317](#)
- oct, [2317](#)
- openmode, [2308](#)
- out, [2317](#)
- precision, [2310](#), [2311](#)
- pwd, [2311](#)
- register\_callback, [2311](#)
- right, [2317](#)
- scientific, [2317](#)
- seekdir, [2308](#)
- setf, [2311](#), [2312](#)
- showbase, [2317](#)
- showpoint, [2317](#)
- showpos, [2318](#)
- skipws, [2318](#)
- sync\_with\_stdio, [2312](#)
- trunc, [2318](#)
- unitbuf, [2318](#)
- unsetf, [2312](#)
- uppercase, [2318](#)
- width, [2313](#)
- xalloc, [2313](#)
- std::ios\_base::failure, [2319](#)
- what, [2319](#)
- std::is\_abstract< \_Tp >, [2320](#)
- std::is\_arithmetic< \_Tp >, [2321](#)
- std::is\_array< typename >, [2321](#)
- std::is\_assignable< \_Tp, \_Up >, [2322](#)
- std::is\_base\_of< \_Base, \_Derived >, [2323](#)
- std::is\_bind\_expression< \_Bind< \_Signature > >, [2325](#)
- std::is\_bind\_expression< \_Bind\_result< \_Result, \_Signature > >, [2326](#)
- std::is\_bind\_expression< \_Tp >, [2324](#)
- std::is\_bind\_expression< const \_Bind< \_Signature > >, [2327](#)
- std::is\_bind\_expression< const \_Bind\_result< \_Result, \_Signature > >, [2328](#)
- std::is\_bind\_expression< const volatile \_Bind< \_Signature > >, [2329](#)
- std::is\_bind\_expression< const volatile \_Bind\_result< \_Result, \_Signature > >, [2330](#)
- std::is\_bind\_expression< volatile \_Bind< \_Signature > >, [2331](#)
- std::is\_bind\_expression< volatile \_Bind\_result< \_Result, \_Signature > >, [2332](#)
- std::is\_class< \_Tp >, [2333](#)
- std::is\_compound< \_Tp >, [2334](#)
- std::is\_const< typename >, [2335](#)
- std::is\_constructible< \_Tp, \_Args >, [2336](#)
- std::is\_convertible< \_From, \_To >, [2337](#)
- std::is\_copy\_assignable< \_Tp >, [2338](#)
- std::is\_copy\_constructible< \_Tp >, [2338](#)
- std::is\_default\_constructible< \_Tp >, [2339](#)
- std::is\_destructible< \_Tp >, [2340](#)
- std::is\_empty< \_Tp >, [2341](#)
- std::is\_enum< \_Tp >, [2342](#)
- std::is\_error\_code\_enum< \_Tp >, [2343](#)
- std::is\_error\_code\_enum< future\_errc >, [2344](#)
- std::is\_error\_condition\_enum< \_Tp >, [2345](#)
- std::is\_floating\_point< \_Tp >, [2346](#)
- std::is\_function< typename >, [2347](#)
- std::is\_fundamental< \_Tp >, [2347](#)
- std::is\_integral< \_Tp >, [2348](#)
- std::is\_literal\_type< \_Tp >, [2349](#)
- std::is\_lvalue\_reference< typename >, [2350](#)
- std::is\_member\_function\_pointer< \_Tp >, [2351](#)
- std::is\_member\_object\_pointer< \_Tp >, [2351](#)
- std::is\_member\_pointer< \_Tp >, [2352](#)
- std::is\_move\_assignable< \_Tp >, [2353](#)
- std::is\_move\_constructible< \_Tp >, [2353](#)
- std::is\_nothrow\_assignable< \_Tp, \_Up >, [2354](#)
- std::is\_nothrow\_constructible< \_Tp, \_Args >, [2354](#)
- std::is\_nothrow\_copy\_assignable< \_Tp >, [2355](#)
- std::is\_nothrow\_copy\_constructible< \_Tp >, [2355](#)
- std::is\_nothrow\_default\_constructible< \_Tp >, [2355](#)
- std::is\_nothrow\_destructible< \_Tp >, [2356](#)
- std::is\_nothrow\_move\_assignable< \_Tp >, [2357](#)
- std::is\_nothrow\_move\_constructible< \_Tp >, [2357](#)
- std::is\_object< \_Tp >, [2358](#)
- std::is\_placeholder< \_Placeholder< \_Num > >, [2360](#)
- std::is\_placeholder< \_Tp >, [2359](#)
- std::is\_pod< \_Tp >, [2361](#)
- std::is\_pointer< \_Tp >, [2362](#)
- std::is\_polymorphic< \_Tp >, [2363](#)
- std::is\_reference< \_Tp >, [2364](#)
- std::is\_rvalue\_reference< typename >, [2364](#)
- std::is\_same< typename, typename >, [2365](#)
- std::is\_scalar< \_Tp >, [2366](#)
- std::is\_signed< \_Tp >, [2366](#)
- std::is\_standard\_layout< \_Tp >, [2368](#)
- std::is\_trivial< \_Tp >, [2369](#)

`std::is_trivially_destructible< _Tp >`, 2370  
`std::is_union< _Tp >`, 2370  
`std::is_unsigned< _Tp >`, 2371  
`std::is_void< _Tp >`, 2371  
`std::is_volatile< typename >`, 2372  
`std::istream_iterator`  
    `difference_type`, 2374  
    `istream_iterator`, 2375  
    `iterator_category`, 2374  
    `pointer`, 2374  
    `reference`, 2374  
    `value_type`, 2375  
`std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`, 2373  
`std::istreambuf_iterator`  
    `char_type`, 2377  
    `difference_type`, 2377  
    `equal`, 2378  
    `int_type`, 2377  
    `istream_type`, 2377  
    `istreambuf_iterator`, 2378  
    `iterator_category`, 2377  
    `operator*`, 2379  
    `operator++`, 2379  
    `pointer`, 2377  
    `reference`, 2377  
    `streambuf_type`, 2378  
    `traits_type`, 2378  
    `value_type`, 2378  
`std::istreambuf_iterator< _CharT, _Traits >`, 2375  
`std::iterator`  
    `difference_type`, 2381  
    `iterator_category`, 2381  
    `pointer`, 2381  
    `reference`, 2381  
    `value_type`, 2381  
`std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`, 2380  
`std::iterator_traits< _Tp * >`, 2381  
`std::iterator_traits< const _Tp * >`, 2382  
`std::length_error`, 2383  
    `what`, 2383  
`std::less`  
    `first_argument_type`, 2384  
    `result_type`, 2384  
    `second_argument_type`, 2385  
`std::less< _Tp >`, 2384  
`std::less_equal`  
    `first_argument_type`, 2386  
    `result_type`, 2386  
    `second_argument_type`, 2386  
`std::less_equal< _Tp >`, 2385  
`std::linear_congruential_engine`  
    `discard`, 2388  
    `increment`, 2390  
    `linear_congruential_engine`, 2387, 2388  
    `max`, 2388  
    `min`, 2388  
    `modulus`, 2390  
    `multiplier`, 2390  
    `operator<<`, 2389  
    `operator>>`, 2390  
    `operator()`, 2388  
    `operator==`, 2389  
    `result_type`, 2387  
    `seed`, 2388, 2389  
`std::list`  
    `_M_create_node`, 2396  
    `assign`, 2396, 2397  
    `back`, 2397  
    `begin`, 2398  
    `cbegin`, 2398  
    `cend`, 2398  
    `clear`, 2398  
    `crbegin`, 2398  
    `crend`, 2398  
    `emplace`, 2399  
    `empty`, 2399  
    `end`, 2399  
    `erase`, 2399, 2400  
    `front`, 2400  
    `get_allocator`, 2400  
    `insert`, 2401, 2402  
    `list`, 2394d  
    `max_size`, 2402  
    `merge`, 2402  
    `operator=`, 2403  
    `pop_back`, 2404  
    `pop_front`, 2404  
    `push_back`, 2404  
    `push_front`, 2404  
    `rbegin`, 2404, 2405  
    `remove`, 2405  
    `remove_if`, 2405  
    `rend`, 2405  
    `resize`, 2406  
    `reverse`, 2406  
    `size`, 2406  
    `sort`, 2406  
    `splice`, 2407  
    `swap`, 2408  
    `unique`, 2408  
`std::list< _Tp, _Alloc >`, 2391  
`std::locale`, 2408  
    `~locale`, 2411  
    `all`, 2415  
    `category`, 2410  
    `classic`, 2412  
    `collate`, 2415

- combine, [2412](#)
- ctype, [2415](#)
- global, [2412](#)
- has\_facet, [2414](#)
- locale, [2410](#), [2411](#)
- messages, [2415](#)
- monetary, [2415](#)
- name, [2412](#)
- none, [2416](#)
- numeric, [2416](#)
- operator(), [2413](#)
- operator=, [2413](#)
- operator==, [2414](#)
- time, [2416](#)
- use\_facet, [2414](#)
- std::locale::facet, [2417](#)
  - ~facet, [2418](#)
  - facet, [2418](#)
- std::locale::id, [2418](#)
  - has\_facet, [2419](#)
  - id, [2419](#)
  - use\_facet, [2419](#)
- std::lock\_guard< \_Mutex >, [2420](#)
- std::logic\_error, [2421](#)
  - logic\_error, [2421](#)
  - what, [2421](#)
- std::logical\_and
  - first\_argument\_type, [2423](#)
  - result\_type, [2423](#)
  - second\_argument\_type, [2423](#)
- std::logical\_and< \_Tp >, [2422](#)
- std::logical\_not
  - argument\_type, [2424](#)
  - result\_type, [2424](#)
- std::logical\_not< \_Tp >, [2423](#)
- std::logical\_or
  - first\_argument\_type, [2425](#)
  - result\_type, [2425](#)
  - second\_argument\_type, [2426](#)
- std::logical\_or< \_Tp >, [2425](#)
- std::lognormal\_distribution
  - max, [2427](#)
  - min, [2427](#)
  - operator<=, [2428](#)
  - operator>=, [2429](#)
  - operator(), [2427](#)
  - operator==, [2429](#)
  - param, [2428](#)
  - reset, [2428](#)
  - result\_type, [2427](#)
- std::lognormal\_distribution< \_RealType >, [2426](#)
- std::lognormal\_distribution< \_RealType >::param\_type,  
[2429](#)
- std::make\_signed< \_Tp >, [2430](#)
- std::make\_unsigned< \_Tp >, [2430](#)
- std::map
  - at, [2435](#)
  - begin, [2435](#)
  - cbegin, [2435](#)
  - cend, [2436](#)
  - clear, [2436](#)
  - count, [2436](#)
  - crbegin, [2436](#)
  - crend, [2436](#)
  - emplace, [2437](#)
  - emplace\_hint, [2437](#)
  - empty, [2437](#)
  - end, [2438](#)
  - equal\_range, [2438](#)
  - erase, [2439](#), [2440](#)
  - find, [2440](#)
  - get\_allocator, [2441](#)
  - insert, [2441](#), [2442](#)
  - key\_comp, [2442](#)
  - lower\_bound, [2442](#), [2443](#)
  - map, [2433](#), [2434](#)
  - max\_size, [2443](#)
  - operator=, [2443](#), [2444](#)
  - rbegin, [2444](#), [2445](#)
  - rend, [2445](#)
  - size, [2445](#)
  - swap, [2445](#)
  - upper\_bound, [2446](#)
  - value\_comp, [2446](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2431](#)
- std::mask\_array< \_Tp >, [2446](#)
- std::match\_results
  - ~match\_results, [2453](#)
  - begin, [2453](#)
  - cbegin, [2453](#)
  - cend, [2453](#)
  - empty, [2453](#)
  - end, [2454](#)
  - format, [2454](#)
  - get\_allocator, [2455](#)
  - length, [2455](#)
  - match\_results, [2452](#), [2453](#)
  - max\_size, [2455](#)
  - operator=, [2455](#)
  - position, [2456](#)
  - prefix, [2456](#)
  - ready, [2456](#)
  - size, [2457](#)
  - str, [2457](#)
  - suffix, [2457](#)
  - swap, [2457](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [2448](#)
- std::mem\_fun1\_ref\_t

- first\_argument\_type, 2459
- result\_type, 2459
- second\_argument\_type, 2459
- std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2458
- std::mem\_fun1\_t
  - first\_argument\_type, 2460
  - result\_type, 2460
  - second\_argument\_type, 2461
- std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2460
- std::mem\_fun\_ref\_t
  - argument\_type, 2462
  - result\_type, 2462
- std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 2461
- std::mem\_fun\_t
  - argument\_type, 2463
  - result\_type, 2463
- std::mem\_fun\_t< \_Ret, \_Tp >, 2462
- std::mersenne\_twister\_engine
  - discard, 2466
  - max, 2466
  - mersenne\_twister\_engine, 2465
  - min, 2466
  - operator<<, 2466
  - operator>>, 2467
  - operator==, 2466
  - result\_type, 2465
- std::messages
  - ~messages, 2470
  - char\_type, 2469
  - id, 2470
  - messages, 2470
  - string\_type, 2469
- std::messages< \_CharT >, 2468
- std::messages\_base, 2471
- std::messages\_byname
  - id, 2473
- std::messages\_byname< \_CharT >, 2472
- std::minus
  - first\_argument\_type, 2474
  - result\_type, 2474
  - second\_argument\_type, 2475
- std::minus< \_Tp >, 2474
- std::modulus
  - first\_argument\_type, 2476
  - result\_type, 2476
  - second\_argument\_type, 2476
- std::modulus< \_Tp >, 2475
- std::money\_base, 2476
- std::money\_get
  - ~money\_get, 2479
  - char\_type, 2479
  - do\_get, 2479, 2480
  - get, 2480
  - id, 2481
  - iter\_type, 2479
  - money\_get, 2479
  - string\_type, 2479
- std::money\_get< \_CharT, \_InIter >, 2477
- std::money\_put
  - ~money\_put, 2483
  - char\_type, 2483
  - do\_put, 2484
  - id, 2486
  - iter\_type, 2483
  - money\_put, 2483
  - put, 2485
  - string\_type, 2483
- std::money\_put< \_CharT, \_OutIter >, 2482
- std::moneypunct
  - ~moneypunct, 2489
  - char\_type, 2488
  - curr\_symbol, 2489
  - decimal\_point, 2489
  - do\_curr\_symbol, 2489
  - do\_decimal\_point, 2490
  - do\_frac\_digits, 2490
  - do\_grouping, 2490
  - do\_neg\_format, 2491
  - do\_negative\_sign, 2491
  - do\_pos\_format, 2491
  - do\_positive\_sign, 2492
  - do\_thousands\_sep, 2492
  - frac\_digits, 2492
  - grouping, 2492
  - id, 2495
  - intl, 2495
  - moneypunct, 2488, 2489
  - neg\_format, 2493
  - negative\_sign, 2493
  - pos\_format, 2494
  - positive\_sign, 2494
  - string\_type, 2488
  - thousands\_sep, 2494
- std::moneypunct< \_CharT, \_Intl >, 2486
- std::moneypunct\_byname
  - curr\_symbol, 2497
  - decimal\_point, 2497
  - do\_curr\_symbol, 2497
  - do\_decimal\_point, 2498
  - do\_frac\_digits, 2498
  - do\_grouping, 2498
  - do\_neg\_format, 2499
  - do\_negative\_sign, 2499
  - do\_pos\_format, 2499
  - do\_positive\_sign, 2500
  - do\_thousands\_sep, 2500
  - frac\_digits, 2500
  - grouping, 2501

- id, 2503
- neg\_format, 2501
- negative\_sign, 2501
- pos\_format, 2502
- positive\_sign, 2502
- thousands\_sep, 2503
- std::moneypunct\_byname<\_CharT, \_Intl >, 2495
- std::move\_iterator<\_Iterator >, 2503
- std::multimap
  - begin, 2508
  - cbegin, 2509
  - cend, 2509
  - clear, 2509
  - count, 2509
  - crbegin, 2510
  - crend, 2510
  - emplace, 2510
  - emplace\_hint, 2510
  - empty, 2511
  - end, 2511
  - equal\_range, 2511, 2512
  - erase, 2512, 2513
  - find, 2513
  - get\_allocator, 2514
  - insert, 2514, 2515
  - key\_comp, 2515
  - lower\_bound, 2515, 2516
  - max\_size, 2516
  - multimap, 2507, 2508
  - operator=, 2516, 2517
  - rbegin, 2517
  - rend, 2517, 2518
  - size, 2518
  - swap, 2518
  - upper\_bound, 2518
  - value\_comp, 2519
- std::multimap<\_Key, \_Tp, \_Compare, \_Alloc >, 2504
- std::multiplies
  - first\_argument\_type, 2520
  - result\_type, 2520
  - second\_argument\_type, 2520
- std::multiplies<\_Tp >, 2519
- std::multiset
  - begin, 2524
  - cbegin, 2524
  - cend, 2525
  - clear, 2525
  - count, 2525
  - crbegin, 2525
  - crend, 2525
  - emplace, 2525
  - emplace\_hint, 2526
  - empty, 2526
  - end, 2526
  - equal\_range, 2527
  - erase, 2527, 2528
  - find, 2529
  - get\_allocator, 2529
  - insert, 2529, 2530
  - key\_comp, 2531
  - lower\_bound, 2531
  - max\_size, 2531
  - multiset, 2523, 2524
  - operator=, 2532
  - rbegin, 2532
  - rend, 2532
  - size, 2533
  - swap, 2533
  - upper\_bound, 2533
  - value\_comp, 2534
- std::multiset<\_Key, \_Compare, \_Alloc >, 2520
- std::mutex, 2534
- std::negate
  - argument\_type, 2535
  - result\_type, 2535
- std::negate<\_Tp >, 2535
- std::negative\_binomial\_distribution
  - k, 2537
  - max, 2537
  - min, 2537
  - operator<=, 2538
  - operator>, 2539
  - operator(), 2537
  - operator==, 2539
  - p, 2538
  - param, 2538
  - reset, 2538
  - result\_type, 2537
- std::negative\_binomial\_distribution<\_IntType >, 2536
- std::negative\_binomial\_distribution<\_IntType >::param\_type, 2539
- std::nested\_exception, 2540
- std::normal\_distribution
  - max, 2542
  - mean, 2542
  - min, 2542
  - normal\_distribution, 2542
  - operator<=, 2543
  - operator>, 2544
  - operator(), 2542, 2543
  - operator==, 2544
  - param, 2543
  - reset, 2543
  - result\_type, 2542
  - stddev, 2543
- std::normal\_distribution<\_RealType >, 2540
- std::normal\_distribution<\_RealType >::param\_type, 2544

std::not\_equal\_to  
     first\_argument\_type, 2546  
     result\_type, 2546  
     second\_argument\_type, 2546  
 std::not\_equal\_to< \_Tp >, 2545  
 std::num\_get  
     ~num\_get, 2549  
     char\_type, 2548  
     do\_get, 2549d  
     get, 2554d  
     id, 2560  
     iter\_type, 2548  
     num\_get, 2548  
 std::num\_get< \_CharT, \_InIter >, 2546  
 std::num\_put  
     ~num\_put, 2562  
     char\_type, 2562  
     do\_put, 2563d  
     id, 2570  
     iter\_type, 2562  
     num\_put, 2562  
     put, 2566d  
 std::num\_put< \_CharT, \_OutIter >, 2560  
 std::numeric\_limits  
     denorm\_min, 2572  
     digits, 2573  
     digits10, 2573  
     epsilon, 2572  
     has\_denorm, 2574  
     has\_denorm\_loss, 2574  
     has\_infinity, 2574  
     has\_quiet\_NaN, 2574  
     has\_signaling\_NaN, 2574  
     infinity, 2572  
     is\_bounded, 2574  
     is\_exact, 2574  
     is\_iec559, 2574  
     is\_integer, 2574  
     is\_modulo, 2574  
     is\_signed, 2575  
     is\_specialized, 2575  
     lowest, 2572  
     max, 2572  
     max\_digits10, 2575  
     max\_exponent, 2575  
     max\_exponent10, 2575  
     min, 2573  
     min\_exponent, 2575  
     min\_exponent10, 2575  
     quiet\_NaN, 2573  
     radix, 2575  
     round\_error, 2573  
     round\_style, 2575  
     signaling\_NaN, 2573  
     tinyness\_before, 2576  
     traps, 2576  
 std::numeric\_limits< \_Tp >, 2571  
 std::numeric\_limits< bool >, 2576  
 std::numeric\_limits< char >, 2577  
 std::numeric\_limits< char16\_t >, 2578  
 std::numeric\_limits< char32\_t >, 2579  
 std::numeric\_limits< double >, 2580  
 std::numeric\_limits< float >, 2581  
 std::numeric\_limits< int >, 2582  
 std::numeric\_limits< long >, 2583  
 std::numeric\_limits< long double >, 2584  
 std::numeric\_limits< long long >, 2585  
 std::numeric\_limits< short >, 2586  
 std::numeric\_limits< signed char >, 2587  
 std::numeric\_limits< unsigned char >, 2588  
 std::numeric\_limits< unsigned int >, 2589  
 std::numeric\_limits< unsigned long >, 2590  
 std::numeric\_limits< unsigned long long >, 2591  
 std::numeric\_limits< unsigned short >, 2592  
 std::numeric\_limits< wchar\_t >, 2593  
 std::numpunct  
     ~numpunct, 2597  
     char\_type, 2596  
     decimal\_point, 2597  
     do\_decimal\_point, 2598  
     do\_falsename, 2598  
     do\_grouping, 2598  
     do\_thousands\_sep, 2598  
     do\_truename, 2599  
     falsename, 2599  
     grouping, 2599  
     id, 2600  
     numpunct, 2597  
     string\_type, 2596  
     thousands\_sep, 2599  
     truename, 2600  
 std::numpunct< \_CharT >, 2595  
 std::numpunct\_byname  
     decimal\_point, 2602  
     do\_decimal\_point, 2602  
     do\_falsename, 2602  
     do\_grouping, 2603  
     do\_thousands\_sep, 2603  
     do\_truename, 2603  
     falsename, 2603  
     grouping, 2604  
     id, 2605  
     thousands\_sep, 2604  
     truename, 2604  
 std::numpunct\_byname< \_CharT >, 2601  
 std::once\_flag, 2605  
     call\_once, 2606  
     once\_flag, 2605

- operator=, 2605
- std::ostream\_iterator
  - char\_type, 2607
  - difference\_type, 2607
  - iterator\_category, 2607
  - operator=, 2609
  - ostream\_iterator, 2608
  - ostream\_type, 2607
  - pointer, 2608
  - reference, 2608
  - traits\_type, 2608
  - value\_type, 2608
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2606
- std::ostreambuf\_iterator
  - char\_type, 2610
  - difference\_type, 2610
  - failed, 2612
  - iterator\_category, 2610
  - operator\*, 2612
  - operator++, 2612
  - operator=, 2612
  - ostream\_type, 2611
  - ostreambuf\_iterator, 2611
  - pointer, 2611
  - reference, 2611
  - streambuf\_type, 2611
  - traits\_type, 2611
  - value\_type, 2611
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 2609
- std::out\_of\_range, 2613
  - what, 2613
- std::output\_iterator\_tag, 2613
- std::overflow\_error, 2614
  - what, 2614
- std::owner\_less< shared\_ptr< \_Tp > >, 2615
  - first\_argument\_type, 2615
  - result\_type, 2615
  - second\_argument\_type, 2615
- std::owner\_less< weak\_ptr< \_Tp > >, 2615
  - first\_argument\_type, 2616
  - result\_type, 2616
  - second\_argument\_type, 2616
- std::pair
  - first, 2619
  - pair, 2618
  - second, 2619
  - second\_type, 2618
- std::pair< \_T1, \_T2 >, 2617
- std::piecewise\_constant\_distribution
  - densities, 2620
  - intervals, 2620
  - max, 2621
  - min, 2621
  - operator<=, 2622
  - operator>=, 2622
  - operator(), 2621
  - operator==, 2622
  - param, 2621
  - reset, 2621
  - result\_type, 2620
- std::piecewise\_constant\_distribution< \_RealType >, 2619
- std::piecewise\_constant\_distribution< \_RealType >::param\_type, 2622
- std::piecewise\_construct\_t, 2623
- std::piecewise\_linear\_distribution
  - densities, 2625
  - intervals, 2625
  - max, 2625
  - min, 2625
  - operator<=, 2626
  - operator>=, 2626
  - operator(), 2625
  - operator==, 2626
  - param, 2625, 2626
  - reset, 2626
  - result\_type, 2625
- std::piecewise\_linear\_distribution< \_RealType >, 2623
- std::piecewise\_linear\_distribution< \_RealType >::param\_type, 2627
- std::placeholders, 609
- std::plus
  - first\_argument\_type, 2628
  - result\_type, 2628
  - second\_argument\_type, 2629
- std::plus< \_Tp >, 2628
- std::pointer\_to\_binary\_function
  - first\_argument\_type, 2630
  - result\_type, 2630
  - second\_argument\_type, 2630
- std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, 2629
- std::pointer\_to\_unary\_function
  - argument\_type, 2631
  - result\_type, 2631
- std::pointer\_to\_unary\_function< \_Arg, \_Result >, 2630
- std::pointer\_traits
  - difference\_type, 2632
  - element\_type, 2632
  - pointer, 2632
- std::pointer\_traits< \_Ptr >, 2631
- std::pointer\_traits< \_Tp \* >, 2632
  - difference\_type, 2633
  - element\_type, 2633
  - pointer, 2633
  - pointer\_to, 2633
- std::poisson\_distribution
  - max, 2635
  - mean, 2635

- min, 2635
- operator<=, 2636
- operator>=, 2636
- operator(), 2635
- operator==, 2636
- param, 2635, 2636
- reset, 2636
- result\_type, 2635
- std::poisson\_distribution< \_IntType >, 2633
- std::poisson\_distribution< \_IntType >::param\_type, 2637
- std::priority\_queue
  - empty, 2639
  - pop, 2639
  - priority\_queue, 2639
  - push, 2640
  - size, 2640
  - top, 2640
- std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 2637
- std::queue
  - back, 2642
  - c, 2643
  - empty, 2642
  - front, 2642, 2643
  - pop, 2643
  - push, 2643
  - queue, 2642
  - size, 2643
- std::queue< \_Tp, \_Sequence >, 2640
- std::random\_access\_iterator\_tag, 2644
- std::random\_device, 2644
  - result\_type, 2645
- std::range\_error, 2645
  - what, 2646
- std::rank< typename >, 2646
- std::ratio< \_Num, \_Den >, 2647
- std::ratio\_equal< \_R1, \_R2 >, 2648
- std::ratio\_not\_equal< \_R1, \_R2 >, 2649
- std::raw\_storage\_iterator
  - difference\_type, 2651
  - iterator\_category, 2651
  - pointer, 2651
  - reference, 2651
  - value\_type, 2651
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 2650
- std::recursive\_mutex, 2651
- std::reference\_wrapper< \_Tp >, 2652
- std::regex\_constants, 610
  - \_\_match\_flag, 612
  - \_\_syntax\_option, 612
  - awk, 613
  - basic, 614
  - collate, 614
  - ECMAScript, 614
  - egrep, 614
  - error\_backref, 612
  - error\_badbrace, 612
  - error\_badrepeat, 613
  - error\_brace, 613
  - error\_brack, 613
  - error\_collate, 613
  - error\_complexity, 613
  - error\_ctype, 613
  - error\_escape, 613
  - error\_paren, 613
  - error\_range, 613
  - error\_space, 613
  - error\_stack, 613
  - error\_type, 612
  - extended, 614
  - format\_default, 614
  - format\_first\_only, 615
  - format\_no\_copy, 615
  - format\_sed, 615
  - grep, 615
  - icase, 615
  - match\_any, 615
  - match\_continuous, 615
  - match\_default, 615
  - match\_flag\_type, 612
  - match\_not\_bol, 616
  - match\_not\_bow, 616
  - match\_not\_eol, 616
  - match\_not\_eow, 616
  - match\_not\_null, 616
  - match\_prev\_avail, 616
  - nosubs, 616
  - optimize, 616
  - syntax\_option\_type, 612
- std::regex\_error, 2653
  - code, 2654
  - regex\_error, 2654
  - what, 2654
- std::regex\_iterator
  - operator\*, 2656
  - operator++, 2656
  - operator->, 2656
  - operator=, 2656
  - operator==, 2656
  - regex\_iterator, 2655
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2654
- std::regex\_token\_iterator
  - operator\*, 2659
  - operator++, 2659, 2660
  - operator->, 2660
  - operator=, 2660
  - operator==, 2660
  - regex\_token\_iterator, 2658, 2659

`std::regex_traits`  
    `getloc`, 2662  
    `imbue`, 2662  
    `length`, 2662  
    `lookup_classname`, 2662  
    `lookup_collatename`, 2663  
    `regex_traits`, 2662  
    `transform`, 2664  
    `transform_primary`, 2664  
    `translate`, 2664  
    `translate_nocase`, 2665  
`std::regex_traits< _Ch_type >`, 2660  
`std::rel_ops`, 616  
    `operator<=`, 617  
    `operator>`, 617  
    `operator>=`, 618  
`std::remove_all_extents< _Tp >`, 2665  
`std::remove_const< _Tp >`, 2666  
`std::remove_cv< _Tp >`, 2666  
`std::remove_extent< _Tp >`, 2667  
`std::remove_pointer< _Tp >`, 2667  
`std::remove_reference< _Tp >`, 2667  
`std::remove_volatile< _Tp >`, 2668  
`std::reverse_iterator`  
    `base`, 2670  
    `iterator_category`, 2670  
    `operator*`, 2671  
    `operator+`, 2671  
    `operator++`, 2671  
    `operator+=`, 2671  
    `operator-`, 2671  
    `operator->`, 2672  
    `operator--`, 2672  
    `operator-=`, 2672  
    `reverse_iterator`, 2670  
    `value_type`, 2670  
`std::reverse_iterator< _Iterator >`, 2668  
`std::runtime_error`, 2673  
    `runtime_error`, 2673  
    `what`, 2674  
`std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >`, 2674  
`std::seed_seq`, 2676  
    `result_type`, 2676  
    `seed_seq`, 2676  
`std::set`  
    `allocator_type`, 2679  
    `begin`, 2683  
    `cbegin`, 2683  
    `cend`, 2683  
    `clear`, 2683  
    `const_iterator`, 2679  
    `const_pointer`, 2679  
    `const_reference`, 2680  
    `const_reverse_iterator`, 2680  
    `count`, 2683  
    `crbegin`, 2684  
    `crend`, 2684  
    `difference_type`, 2680  
    `emplace`, 2684  
    `emplace_hint`, 2684  
    `empty`, 2685  
    `end`, 2685  
    `equal_range`, 2685  
    `erase`, 2686  
    `find`, 2687  
    `get_allocator`, 2687  
    `insert`, 2688, 2689  
    `iterator`, 2680  
    `key_comp`, 2689  
    `key_compare`, 2680  
    `key_type`, 2680  
    `lower_bound`, 2689  
    `max_size`, 2690  
    `operator=`, 2690  
    `pointer`, 2680  
    `rbegin`, 2690  
    `reference`, 2680  
    `rend`, 2691  
    `reverse_iterator`, 2681  
    `set`, 2681, 2682  
    `size`, 2691  
    `size_type`, 2681  
    `swap`, 2691  
    `upper_bound`, 2691  
    `value_comp`, 2692  
    `value_compare`, 2681  
    `value_type`, 2681  
`std::set< _Key, _Compare, _Alloc >`, 2677  
`std::shared_ptr`  
    `allocate_shared`, 2698  
    `shared_ptr`, 2694d  
`std::shared_ptr< _Tp >`, 2692  
`std::shuffle_order_engine`  
    `base`, 2701  
    `discard`, 2701  
    `max`, 2701  
    `min`, 2701  
    `operator<<`, 2702  
    `operator>>`, 2703  
    `operator()`, 2701  
    `operator==`, 2702  
    `result_type`, 2700  
    `seed`, 2701, 2702  
    `shuffle_order_engine`, 2700  
`std::shuffle_order_engine< _RandomNumberEngine, __k >`, 2698  
`std::slice`, 2703

std::slice\_array< \_Tp >, 2704  
 std::stack  
   empty, 2707  
   pop, 2707  
   push, 2707  
   size, 2707  
   stack, 2707  
   top, 2707  
 std::stack< \_Tp, \_Sequence >, 2705  
 std::student\_t\_distribution  
   max, 2709  
   min, 2709  
   operator<=, 2710  
   operator>=, 2710  
   operator(), 2709  
   operator==, 2710  
   param, 2709, 2710  
   reset, 2710  
   result\_type, 2709  
 std::student\_t\_distribution< \_RealType >, 2708  
 std::student\_t\_distribution< \_RealType >::param\_type, 2711  
 std::sub\_match  
   compare, 2713, 2714  
   first, 2715  
   length, 2714  
   operator string\_type, 2714  
   second, 2715  
   second\_type, 2713  
   str, 2714  
 std::sub\_match< \_Biter >, 2712  
 std::system\_error, 2715  
   what, 2716  
 std::this\_thread, 618  
   get\_id, 618  
   sleep\_for, 618  
   sleep\_until, 619  
   yield, 619  
 std::thread, 2716  
   native\_handle, 2717  
 std::thread::id, 2717  
 std::time\_base, 2718  
 std::time\_get  
   ~time\_get, 2721  
   char\_type, 2720  
   date\_order, 2721  
   do\_date\_order, 2721  
   do\_get\_date, 2722  
   do\_get\_monthname, 2722  
   do\_get\_time, 2723  
   do\_get\_weekday, 2723  
   do\_get\_year, 2724  
   get\_date, 2724  
   get\_monthname, 2725  
   get\_time, 2725  
   get\_weekday, 2726  
   get\_year, 2726  
   id, 2727  
   iter\_type, 2720  
   time\_get, 2721  
 std::time\_get< \_CharT, \_InIter >, 2719  
 std::time\_get\_byname  
   date\_order, 2729  
   do\_date\_order, 2729  
   do\_get\_date, 2729  
   do\_get\_monthname, 2730  
   do\_get\_time, 2731  
   do\_get\_weekday, 2731  
   do\_get\_year, 2732  
   get\_date, 2732  
   get\_monthname, 2733  
   get\_time, 2733  
   get\_weekday, 2734  
   get\_year, 2734  
   id, 2735  
 std::time\_get\_byname< \_CharT, \_InIter >, 2727  
 std::time\_put  
   ~time\_put, 2737  
   char\_type, 2736  
   do\_put, 2737  
   id, 2739  
   iter\_type, 2736  
   put, 2738  
   time\_put, 2737  
 std::time\_put< \_CharT, \_OutIter >, 2735  
 std::time\_put\_byname  
   do\_put, 2740  
   id, 2742  
   put, 2741  
 std::time\_put\_byname< \_CharT, \_OutIter >, 2739  
 std::tr1, 619  
 std::tr1::\_\_detail, 622  
 std::tr2, 622  
   operator<, 625  
   operator<=, 625  
   operator<=, 625  
   operator>, 626  
   operator>=, 626  
   operator>=, 626  
   operator^, 626  
   operator-, 625  
   operator==, 625  
   operator&, 624  
 std::tr2::\_\_detail, 627  
 std::tr2::\_\_dynamic\_bitset\_base  
   \_M\_w, 2744  
 std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc >, 2742  
 std::tr2::\_\_reflection\_typelist< \_First, \_Rest...>, 2744

std::tr2::\_\_reflection\_typelist<>, 2744  
 std::tr2::bases< \_Tp >, 2745  
 std::tr2::bool\_set, 2745  
     bool\_set, 2746  
     equals, 2746  
     is\_emptyset, 2746  
     is\_indeterminate, 2746  
     is\_singleton, 2746  
     operator bool, 2747  
 std::tr2::direct\_bases< \_Tp >, 2747  
 std::tr2::dynamic\_bitset  
     all, 2753  
     any, 2753  
     append, 2753  
     clear, 2753  
     count, 2753  
     dynamic\_bitset, 2751d  
     empty, 2754  
     find\_first, 2754  
     find\_next, 2754  
     flip, 2754  
     get\_allocator, 2755  
     max\_size, 2755  
     none, 2755  
     num\_blocks, 2755  
     operator<, 2756  
     operator<=, 2756  
     operator>, 2757  
     operator>=, 2757  
     operator~, 2758  
     operator^=, 2758  
     operator-=, 2756  
     operator=, 2756, 2757  
     operator&=, 2755, 2756  
     push\_back, 2758  
     reset, 2759  
     resize, 2759  
     set, 2759  
     size, 2760  
     swap, 2760  
     test, 2760  
     to\_string, 2760  
     to\_ullong, 2760  
     to\_ulong, 2761  
 std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2747  
 std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::reference,  
     2761  
 std::try\_to\_lock\_t, 2762  
 std::tuple< \_Elements >, 2762  
 std::tuple< \_T1, \_T2 >, 2764  
 std::tuple\_element< 0, tuple< \_Head, \_Tail...> >, 2765  
 std::tuple\_element< \_i, tuple< \_Head, \_Tail...> >, 2766  
 std::tuple\_size< tuple< \_Elements...> >, 2767  
 std::type\_index, 2768  
 std::type\_info, 2768  
     ~type\_info, 2769  
     name, 2769  
 std::unary\_function  
     argument\_type, 2771  
     result\_type, 2771  
 std::unary\_function< \_Arg, \_Result >, 2770  
 std::unary\_negate  
     argument\_type, 2773  
     result\_type, 2773  
 std::unary\_negate< \_Predicate >, 2772  
 std::underflow\_error, 2773  
     what, 2774  
 std::underlying\_type< \_Tp >, 2774  
 std::uniform\_int\_distribution  
     max, 2775  
     min, 2775  
     operator(), 2776  
     operator==, 2776  
     param, 2776  
     reset, 2776  
     result\_type, 2775  
     uniform\_int\_distribution, 2775  
 std::uniform\_int\_distribution< \_IntType >, 2774  
 std::uniform\_int\_distribution< \_IntType >::param\_type,  
     2777  
 std::uniform\_real\_distribution  
     max, 2779  
     min, 2779  
     operator(), 2779  
     operator==, 2780  
     param, 2779  
     reset, 2779  
     result\_type, 2778  
     uniform\_real\_distribution, 2778  
 std::uniform\_real\_distribution< \_RealType >, 2777  
 std::uniform\_real\_distribution< \_RealType >::param\_  
     type, 2780  
 std::unique\_lock< \_Mutex >, 2780  
 std::unique\_ptr< \_Tp, \_Dp >, 2781  
 std::unordered\_map  
     allocator\_type, 2786  
     at, 2790  
     begin, 2791  
     bucket\_count, 2792  
     cbegin, 2792  
     cend, 2792  
     clear, 2793  
     const\_iterator, 2787  
     const\_local\_iterator, 2787  
     const\_pointer, 2787  
     const\_reference, 2787  
     count, 2793  
     difference\_type, 2787

- emplace, 2793
- emplace\_hint, 2794
- empty, 2794
- end, 2794, 2795
- equal\_range, 2795
- erase, 2796, 2797
- find, 2797, 2798
- get\_allocator, 2798
- hash\_function, 2798
- hasher, 2787
- insert, 2798d
- iterator, 2787
- key\_eq, 2800
- key\_equal, 2788
- key\_type, 2788
- load\_factor, 2801
- local\_iterator, 2788
- mapped\_type, 2788
- max\_bucket\_count, 2801
- max\_load\_factor, 2801
- max\_size, 2801
- operator=, 2801, 2802
- pointer, 2788
- reference, 2788
- rehash, 2803
- reserve, 2803
- size, 2803
- size\_type, 2788
- swap, 2803
- unordered\_map, 2789, 2790
- value\_type, 2789
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 2784
- std::unordered\_multimap
  - allocator\_type, 2806
  - begin, 2810, 2811
  - bucket\_count, 2811
  - cbegin, 2811
  - cend, 2812
  - clear, 2812
  - const\_iterator, 2806
  - const\_local\_iterator, 2807
  - const\_pointer, 2807
  - const\_reference, 2807
  - count, 2812
  - difference\_type, 2807
  - emplace, 2813
  - emplace\_hint, 2813
  - empty, 2813
  - end, 2814
  - equal\_range, 2814, 2815
  - erase, 2815, 2816
  - find, 2816, 2817
  - get\_allocator, 2817
  - hash\_function, 2817
  - hasher, 2807
  - insert, 2817d
  - iterator, 2807
  - key\_eq, 2820
  - key\_equal, 2807
  - key\_type, 2807
  - load\_factor, 2820
  - local\_iterator, 2808
  - mapped\_type, 2808
  - max\_bucket\_count, 2820
  - max\_load\_factor, 2820
  - max\_size, 2820
  - operator=, 2820, 2821
  - pointer, 2808
  - reference, 2808
  - rehash, 2821
  - reserve, 2821
  - size, 2821
  - size\_type, 2808
  - swap, 2822
  - unordered\_multimap, 2808, 2809
  - value\_type, 2808
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 2804
- std::unordered\_multiset
  - allocator\_type, 2825
  - begin, 2828, 2829
  - bucket\_count, 2829
  - cbegin, 2829
  - cend, 2830
  - clear, 2830
  - const\_iterator, 2825
  - const\_local\_iterator, 2825
  - const\_pointer, 2825
  - const\_reference, 2825
  - count, 2830
  - difference\_type, 2825
  - emplace, 2830
  - emplace\_hint, 2831
  - empty, 2831
  - end, 2831, 2832
  - equal\_range, 2832
  - erase, 2833, 2834
  - find, 2834, 2835
  - get\_allocator, 2835
  - hash\_function, 2835
  - hasher, 2825
  - insert, 2835d
  - iterator, 2825
  - key\_eq, 2837
  - key\_equal, 2826
  - key\_type, 2826
  - load\_factor, 2837

- local\_iterator, [2826](#)
- max\_bucket\_count, [2838](#)
- max\_load\_factor, [2838](#)
- max\_size, [2838](#)
- operator=, [2838](#)
- pointer, [2826](#)
- reference, [2826](#)
- rehash, [2839](#)
- reserve, [2839](#)
- size, [2839](#)
- size\_type, [2826](#)
- swap, [2839](#)
- unordered\_multiset, [2827](#), [2828](#)
- value\_type, [2826](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc  
>, [2822](#)
- std::unordered\_set
  - allocator\_type, [2842](#)
  - begin, [2846](#)
  - bucket\_count, [2847](#)
  - cbegin, [2847](#)
  - cend, [2847](#)
  - clear, [2848](#)
  - const\_iterator, [2842](#)
  - const\_local\_iterator, [2843](#)
  - const\_pointer, [2843](#)
  - const\_reference, [2843](#)
  - count, [2848](#)
  - difference\_type, [2843](#)
  - emplace, [2848](#)
  - emplace\_hint, [2848](#)
  - empty, [2849](#)
  - end, [2849](#), [2850](#)
  - equal\_range, [2850](#)
  - erase, [2851](#), [2852](#)
  - find, [2852](#), [2853](#)
  - get\_allocator, [2853](#)
  - hash\_function, [2853](#)
  - hasher, [2843](#)
  - insert, [2853d](#)
  - iterator, [2843](#)
  - key\_eq, [2856](#)
  - key\_equal, [2843](#)
  - key\_type, [2843](#)
  - load\_factor, [2856](#)
  - local\_iterator, [2844](#)
  - max\_bucket\_count, [2856](#)
  - max\_load\_factor, [2856](#)
  - max\_size, [2856](#)
  - operator=, [2857](#)
  - pointer, [2844](#)
  - reference, [2844](#)
  - rehash, [2857](#)
  - reserve, [2857](#)
  - size, [2858](#)
  - size\_type, [2844](#)
  - swap, [2858](#)
  - unordered\_set, [2844](#), [2845](#)
  - value\_type, [2844](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [2840](#)
- std::uses\_allocator< \_Tp, \_Alloc >, [2859](#)
- std::uses\_allocator< tuple< \_Types...>, \_Alloc >, [2860](#)
- std::valarray
  - valarray, [2863](#)
- std::valarray< \_Tp >, [2861](#)
- std::vector
  - ~vector, [2869](#)
  - \_M\_allocate\_and\_copy, [2870](#)
  - \_M\_range\_check, [2870](#)
  - assign, [2870](#)
  - at, [2871](#)
  - back, [2871](#), [2872](#)
  - begin, [2872](#)
  - capacity, [2872](#)
  - cbegin, [2872](#)
  - cend, [2872](#)
  - clear, [2872](#)
  - crbegin, [2873](#)
  - crend, [2873](#)
  - data, [2873](#)
  - emplace, [2873](#)
  - empty, [2873](#)
  - end, [2874](#)
  - erase, [2874](#)
  - front, [2875](#)
  - insert, [2875](#), [2876](#)
  - max\_size, [2877](#)
  - operator=, [2877](#)
  - pop\_back, [2878](#)
  - push\_back, [2879](#)
  - rbegin, [2879](#)
  - rend, [2879](#)
  - reserve, [2879](#)
  - resize, [2880](#)
  - shrink\_to\_fit, [2880](#)
  - size, [2880](#)
  - swap, [2881](#)
  - vector, [2867d](#)
- std::vector< \_Tp, \_Alloc >, [2864](#)
- std::vector< bool, \_Alloc >, [2881](#)
- std::weak\_ptr< \_Tp >, [2884](#)
- std::weibull\_distribution
  - a, [2886](#)
  - b, [2886](#)
  - max, [2886](#)
  - min, [2887](#)
  - operator(), [2887](#)

- operator==, [2887](#)
- param, [2887](#)
- reset, [2887](#)
- result\_type, [2886](#)
- std::weibull\_distribution< \_RealType >, [2885](#)
- std::weibull\_distribution< \_RealType >::param\_type, [2888](#)
- stdc++.h, [3212](#)
- stddev
  - std::normal\_distribution, [2543](#)
- stdexcept, [3212](#)
- stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf, [766](#)
- stdio\_filebuf.h, [3212](#)
- stdio\_sync\_filebuf.h, [3213](#)
- stdtr1c++.h, [3213](#)
- stl\_algo.h, [3213](#)
- stl\_algobase.h, [3223](#)
- stl\_bvector.h, [3225](#)
- stl\_construct.h, [3226](#)
- stl\_deque.h, [3226](#)
- stl\_function.h, [3229](#)
- stl\_heap.h, [3231](#)
- stl\_iterator.h, [3233](#)
- stl\_iterator\_base\_funcs.h, [3236](#)
- stl\_iterator\_base\_types.h, [3237](#)
- stl\_list.h, [3238](#)
- stl\_map.h, [3239](#)
- stl\_multimap.h, [3239](#)
- stl\_multiset.h, [3240](#)
- stl\_numeric.h, [3241](#)
- stl\_pair.h, [3242](#)
- stl\_queue.h, [3243](#)
- stl\_raw\_storage\_iter.h, [3244](#)
- stl\_relops.h, [3244](#)
- stl\_set.h, [3245](#)
- stl\_stack.h, [3245](#)
- stl\_tempbuf.h, [3246](#)
- stl\_tree.h, [3247](#)
- stl\_uninitialized.h, [3248](#)
- stl\_vector.h, [3249](#)
- str
  - std::basic\_istream, [1797](#), [1798](#)
  - std::basic\_ostringstream, [1895](#), [1896](#)
  - std::basic\_stringbuf, [1987](#), [1988](#)
  - std::basic\_stringstream, [2030](#)
  - std::match\_results, [2457](#)
  - std::sub\_match, [2714](#)
- stream\_iterator.h, [3250](#)
- streambuf, [3251](#)
  - I/O, [44](#)
- streambuf.tcc, [3251](#)
- streambuf\_iterator.h, [3252](#)
- streambuf\_type
  - std::istreambuf\_iterator, [2378](#)
  - std::ostreambuf\_iterator, [2611](#)
- streamoff
  - std, [496](#)
- streampos
  - std, [497](#)
- streamsize
  - std, [497](#)
- stride
  - Numeric Arrays, [102](#)
- string, [3253](#)
  - Strings, [237](#)
- string\_conversions.h, [3256](#)
- string\_type
  - std::collate, [2089](#)
  - std::collate\_byname, [2094](#)
  - std::messages, [2469](#)
  - std::money\_get, [2479](#)
  - std::money\_put, [2483](#)
  - std::moneypunct, [2488](#)
  - std::numpunct, [2596](#)
- stringbuf
  - I/O, [44](#)
- stringfwd.h, [3256](#)
- Strings, [237](#)
  - string, [237](#)
  - u16string, [237](#)
  - u32string, [237](#)
  - wstring, [237](#)
- stringstream
  - I/O, [44](#)
- strstream, [3257](#)
- substr
  - \_\_gnu\_cxx::\_\_versa\_string, [694](#)
  - std::basic\_string, [1974](#)
- subtractive\_rng
  - \_\_gnu\_cxx::subtractive\_rng, [801](#)
- suffix
  - std::match\_results, [2457](#)
- sum
  - Numeric Arrays, [102](#)
- sungetc
  - \_\_gnu\_cxx::enc\_filebuf, [726](#)
  - \_\_gnu\_cxx::stdio\_filebuf, [779](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, [797](#)
  - std::basic\_filebuf, [1564](#)
  - std::basic\_streambuf, [1926](#)
  - std::basic\_stringbuf, [1988](#)
- swap
  - \_\_gnu\_cxx, [312](#)
  - \_\_gnu\_cxx::\_\_versa\_string, [695](#)
  - \_\_gnu\_debug::basic\_string, [872](#)
  - \_\_gnu\_pbds::sample\_probe\_fn, [1206](#)
  - \_\_gnu\_pbds::sample\_range\_hashing, [1207](#)

- \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 1208
- \_\_gnu\_pbds::sample\_resize\_policy, 1211
- \_\_gnu\_pbds::sample\_resize\_trigger, 1214
- \_\_gnu\_pbds::sample\_size\_policy, 1215
- \_\_gnu\_pbds::sample\_update\_policy, 1218
- Mutexes, 53
- Numeric Arrays, 102
- Regular Expressions, 209, 210
- std, 560, 561
- std::\_\_debug, 570
- std::\_\_detail::\_\_Nfa, 1404
- std::\_\_profile, 596
- std::basic\_regex, 1911
- std::basic\_string, 1974
- std::deque, 2193
- std::forward\_list, 2239
- std::function< \_Res(\_ArgTypes...)>, 2250
- std::list, 2408
- std::map, 2445
- std::match\_results, 2457
- std::multimap, 2518
- std::multiset, 2533
- std::set, 2691
- std::tr2::dynamic\_bitset, 2760
- std::unordered\_map, 2803
- std::unordered\_multimap, 2822
- std::unordered\_multiset, 2839
- std::unordered\_set, 2858
- std::vector, 2881
- Utilities, 70
- swap\_ranges
  - Mutating, 118
- sync
  - \_\_gnu\_cxx::enc\_filebuf, 727
  - \_\_gnu\_cxx::stdio\_filebuf, 779
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 797
  - std::basic\_filebuf, 1564
  - std::basic\_fstream, 1608
  - std::basic\_ifstream, 1649
  - std::basic\_iostream, 1718
  - std::basic\_istream, 1757
  - std::basic\_istreamstream, 1798
  - std::basic\_streambuf, 1927
  - std::basic\_stringbuf, 1988
  - std::basic\_stringstream, 2030
- sync\_with\_stdio
  - std::basic\_fstream, 1609
  - std::basic\_ifstream, 1649
  - std::basic\_ios, 1674
  - std::basic\_iostream, 1718
  - std::basic\_istream, 1757
  - std::basic\_istreamstream, 1798
  - std::basic\_ofstream, 1831
  - std::basic\_ostream, 1862
- std::basic\_ostreamstream, 1896
- std::basic\_stringstream, 2031
- std::ios\_base, 2312
- syntax\_option\_type
  - std::regex\_constants, 612
- synth\_access\_traits
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_A-Traits, Node\_Update, pat\_trie\_tag, \_Alloc >, 1166
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_A-Traits, Node\_Update, pat\_trie\_tag, \_Alloc >, 1167
- synth\_access\_traits.hpp, 3257
- system\_error, 3257
- t
  - std::binomial\_distribution, 2052
- TLB\_size
  - \_\_gnu\_parallel::\_Settings, 965
- table
  - std::ctype< char >, 2128
  - std::ctype\_byname< char >, 2163
- table\_size
  - std::ctype< char >, 2130
  - std::ctype\_byname< char >, 2166
- tag\_and\_trait.hpp, 3259
- Tags, 259
  - trivial\_iterator\_difference\_type, 259
- tags.h, 3261
- tan
  - Complex Numbers, 37
- tanh
  - Complex Numbers, 38
- target
  - std::function< \_Res(\_ArgTypes...)>, 2250
- target\_type
  - std::function< \_Res(\_ArgTypes...)>, 2250
- tellg
  - std::basic\_fstream, 1609
  - std::basic\_ifstream, 1650
  - std::basic\_iostream, 1719
  - std::basic\_istream, 1758
  - std::basic\_istreamstream, 1799
  - std::basic\_stringstream, 2031
- tellp
  - std::basic\_fstream, 1609
  - std::basic\_iostream, 1719
  - std::basic\_ofstream, 1832
  - std::basic\_ostream, 1863
  - std::basic\_ostreamstream, 1896
  - std::basic\_stringstream, 2031
- temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer, 803
- terminate

- Exceptions, 26
- terminate\_handler
  - Exceptions, 25
- test
  - std, 561
  - std::tr2::dynamic\_bitset, 2760
- tgmath.h, 3262
- thin\_heap\_.hpp, 3262
- thousands\_sep
  - std::moneypunct, 2494
  - std::moneypunct\_byname, 2503
  - std::numpunct, 2599
  - std::numpunct\_byname, 2604
- thread, 3263
- Threads, 58
- throw\_allocator.h, 3264
- throw\_with\_nested
  - Exceptions, 26
- tie
  - std::basic\_fstream, 1610
  - std::basic\_ifstream, 1650
  - std::basic\_ios, 1674
  - std::basic\_iostream, 1719
  - std::basic\_istream, 1758
  - std::basic\_istreamstringstream, 1799
  - std::basic\_ofstream, 1832
  - std::basic\_ostream, 1863
  - std::basic\_ostreamstringstream, 1896, 1897
  - std::basic\_stringstream, 2031, 2032
  - Utilities, 70
- Time, 27
- time
  - std::locale, 2416
- time\_get
  - std::time\_get, 2721
- time\_members.h, 3265
- time\_point\_cast
  - std::chrono, 600
- time\_put
  - std::time\_put, 2737
- tinyness\_before
  - std::\_\_numeric\_limits\_base, 1423
  - std::numeric\_limits, 2576
- to\_string
  - std, 562
  - std::tr2::dynamic\_bitset, 2760
- to\_ullong
  - std::tr2::dynamic\_bitset, 2760
- to\_ulong
  - std, 562
  - std::tr2::dynamic\_bitset, 2761
- tolower
  - std, 562
  - std::\_\_ctype\_abstract\_base, 1300
  - std::ctype, 2117
  - std::ctype< char >, 2128
  - std::ctype< wchar\_t >, 2140, 2141
  - std::ctype\_byname, 2153
  - std::ctype\_byname< char >, 2163, 2164
- top
  - std::priority\_queue, 2640
  - std::stack, 2707
- toupper
  - std, 562
  - std::\_\_ctype\_abstract\_base, 1301
  - std::ctype, 2118
  - std::ctype< char >, 2128, 2129
  - std::ctype< wchar\_t >, 2141, 2142
  - std::ctype\_byname, 2153, 2154
  - std::ctype\_byname< char >, 2164
- trace\_fn\_imps.hpp, 3266, 3267
- Traits, 263
- traits.hpp, 3267d
- traits\_type
  - std::basic\_ios, 1664
  - std::basic\_istream::sentry, 1766
  - std::basic\_streambuf, 1916
  - std::istreambuf\_iterator, 2378
  - std::ostream\_iterator, 2608
  - std::ostreambuf\_iterator, 2611
- transform
  - Mutating, 118
  - std::collate, 2092
  - std::collate\_byname, 2096
  - std::regex\_traits, 2664
- transform\_minimal\_n
  - \_\_gnu\_parallel::Settings, 965
- transform\_primary
  - std::regex\_traits, 2664
- translate
  - std::regex\_traits, 2664
- translate\_nocase
  - std::regex\_traits, 2665
- traps
  - std::\_\_numeric\_limits\_base, 1424
  - std::numeric\_limits, 2576
- tree
  - \_\_gnu\_pbds::tree, 1223, 1224
  - tree\_metadata\_helper< Node\_Update, \_BTp >, 2888
  - tree\_policy.hpp, 3269
  - tree\_trace\_base.hpp, 3270
  - tree\_traits< Key, Data, Cmp\_Fn, Node\_Update, Tag, \_Alloc >, 2889
- trie
  - \_\_gnu\_pbds::trie, 1229
  - trie\_metadata\_helper< Node\_Update, \_BTp >, 2889
  - trie\_policy.hpp, 3270
  - trie\_policy\_base.hpp, 3271

- trie\_string\_access\_traits\_imp.hpp, [3271](#)
- trie\_traits< Key, Data, \_ATraits, Node\_Update, Tag, \_Alloc >, [2889](#)
- trivial\_iterator\_difference\_type
  - Tags, [259](#)
- true\_type
  - Metaprogramming, [63](#)
- truename
  - std::numpunct, [2600](#)
  - std::numpunct\_byname, [2604](#)
- trunc
  - std::basic\_fstream, [1617](#)
  - std::basic\_ifstream, [1657](#)
  - std::basic\_ios, [1681](#)
  - std::basic\_iostream, [1727](#)
  - std::basic\_istream, [1765](#)
  - std::basic\_istreamstream, [1806](#)
  - std::basic\_ofstream, [1839](#)
  - std::basic\_ostream, [1870](#)
  - std::basic\_ostreamstream, [1903](#)
  - std::basic\_stringstream, [2039](#)
  - std::ios\_base, [2318](#)
- try\_lock
  - Mutexes, [53](#)
- tuple, [3271](#)
- tuple\_cat
  - Utilities, [71](#)
- tuple\_element< \_\_i, \_Tp >, [2890](#)
- tuple\_element< \_Int, \_Tp >, [2890](#)
- tuple\_size< \_Tp >, [2890](#), [2891](#)
- type
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1050](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1050](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI >, [1051](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI >, [1051](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI >, [1052](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI >, [1053](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1054](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1054](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI >, [1055](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_TI >, [1055](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_TI >, [1056](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_TI >, [1057](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V-Tp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, [1047](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V-Tp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, [1047](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V-Tp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, [1048](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, [1049](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_V-Tp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, [1049](#)
  - \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [1057](#)
  - \_\_gnu\_pbds::detail::default\_eq\_fn, [1058](#)
  - \_\_gnu\_pbds::detail::default\_hash\_fn, [1058](#)
  - \_\_gnu\_pbds::detail::default\_probe\_fn, [1059](#)
  - \_\_gnu\_pbds::detail::default\_resize\_policy, [1060](#)
  - \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string< Char, Char\_Traits, std::allocator< char > > >, [1060](#)
  - \_\_gnu\_pbds::detail::default\_update\_policy, [1061](#)
  - \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, [1063](#)
- type\_base< Key, Mapped, \_Alloc, Store\_Hash >, [2891](#)
- type\_traits, [3273](#), [3278](#)
- type\_traits.h, [3278](#)
- type\_utils.hpp, [3279](#)
- typeid, [3279](#)
- typeinfo, [3280](#)
- typelist.h, [3280](#)
- types.h, [3281](#)
- types\_traits.hpp, [3282](#)
- u16streampos
  - std, [497](#)
- u16string
  - Strings, [237](#)
- u32streampos
  - std, [497](#)
- u32string

- Strings, [237](#)
- uflow
  - [\\_\\_gnu\\_cxx::enc\\_filebuf](#), [727](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf](#), [779](#)
  - [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf](#), [797](#)
  - [std::basic\\_filebuf](#), [1564](#)
  - [std::basic\\_streambuf](#), [1927](#)
  - [std::basic\\_stringbuf](#), [1989](#)
- uncaught\_exception
  - Exceptions, [26](#)
- underflow
  - [\\_\\_gnu\\_cxx::enc\\_filebuf](#), [727](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf](#), [779](#)
  - [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf](#), [797](#)
  - [std::basic\\_filebuf](#), [1565](#)
  - [std::basic\\_streambuf](#), [1927](#)
  - [std::basic\\_stringbuf](#), [1989](#)
- unexpected
  - Exceptions, [26](#)
- unexpected\_handler
  - Exceptions, [25](#)
- unget
  - [std::basic\\_fstream](#), [1610](#)
  - [std::basic\\_ifstream](#), [1651](#)
  - [std::basic\\_iostream](#), [1720](#)
  - [std::basic\\_istream](#), [1759](#)
  - [std::basic\\_istreamstream](#), [1800](#)
  - [std::basic\\_stringstream](#), [2032](#)
- Uniform Distributions, [272](#)
  - [operator<<](#), [273](#)
  - [operator>>](#), [273](#), [274](#)
- uniform\_int\_distribution
  - [std::uniform\\_int\\_distribution](#), [2775](#)
- uniform\_real\_distribution
  - [std::uniform\\_real\\_distribution](#), [2778](#)
- uninitialized\_copy
  - [std](#), [563](#)
- uninitialized\_copy\_n
  - SGL, [14](#)
  - [std](#), [563](#)
- uninitialized\_fill
  - [std](#), [563](#)
- uninitialized\_fill\_n
  - [std](#), [564](#)
- unique
  - Mutating, [119](#)
  - [std::forward\\_list](#), [2239](#), [2240](#)
  - [std::list](#), [2408](#)
- unique\_copy
  - Mutating, [120](#)
- [unique\\_copy.h](#), [3283](#)
- [unique\\_copy\\_minimal\\_n](#)
  - [\\_\\_gnu\\_parallel::\\_Settings](#), [965](#)
- [unique\\_ptr.h](#), [3283](#)
- unitbuf
  - [std](#), [564](#)
  - [std::basic\\_fstream](#), [1617](#)
  - [std::basic\\_ifstream](#), [1657](#)
  - [std::basic\\_ios](#), [1681](#)
  - [std::basic\\_iostream](#), [1727](#)
  - [std::basic\\_istream](#), [1765](#)
  - [std::basic\\_istreamstream](#), [1806](#)
  - [std::basic\\_ofstream](#), [1839](#)
  - [std::basic\\_ostream](#), [1870](#)
  - [std::basic\\_ostreamstream](#), [1904](#)
  - [std::basic\\_stringstream](#), [2039](#)
  - [std::ios\\_base](#), [2318](#)
- Unordered Associative, [20](#)
- [unordered\\_map](#), [3284d](#)
  - [std::unordered\\_map](#), [2789](#), [2790](#)
- [unordered\\_map.h](#), [3287](#)
- [unordered\\_multimap](#)
  - [std::unordered\\_multimap](#), [2808](#), [2809](#)
- [unordered\\_multiset](#)
  - [std::unordered\\_multiset](#), [2827](#), [2828](#)
- [unordered\\_set](#), [3288](#), [3289](#)
  - [std::unordered\\_set](#), [2844](#), [2845](#)
- [unordered\\_set.h](#), [3290](#)
- unsetf
  - [std::basic\\_fstream](#), [1611](#)
  - [std::basic\\_ifstream](#), [1651](#)
  - [std::basic\\_ios](#), [1675](#)
  - [std::basic\\_iostream](#), [1720](#)
  - [std::basic\\_istream](#), [1759](#)
  - [std::basic\\_istreamstream](#), [1800](#)
  - [std::basic\\_ofstream](#), [1833](#)
  - [std::basic\\_ostream](#), [1864](#)
  - [std::basic\\_ostreamstream](#), [1897](#)
  - [std::basic\\_stringstream](#), [2032](#)
  - [std::ios\\_base](#), [2312](#)
- unshift
  - [std::\\_\\_codecvt\\_abstract\\_base](#), [1290](#)
  - [std::codecvt](#), [2071](#)
  - [std::codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#), [2075](#)
  - [std::codecvt< char, char, mbstate\\_t >](#), [2078](#)
  - [std::codecvt< wchar\\_t, char, mbstate\\_t >](#), [2082](#)
  - [std::codecvt\\_byname](#), [2087](#)
- [update\\_fn\\_imps.hpp](#), [3291](#)
- upper\_bound
  - Binary Search, [159](#)
  - [std::map](#), [2446](#)
  - [std::multimap](#), [2518](#)
  - [std::multiset](#), [2533](#)
  - [std::set](#), [2691](#)
- uppercase
  - [std](#), [564](#)
  - [std::basic\\_fstream](#), [1617](#)

- std::basic\_ifstream, 1657
- std::basic\_ios, 1681
- std::basic\_iostream, 1727
- std::basic\_istream, 1766
- std::basic\_istreamstream, 1806
- std::basic\_ofstream, 1839
- std::basic\_ostream, 1870
- std::basic\_ostreamstream, 1904
- std::basic\_stringstream, 2039
- std::ios\_base, 2318
- use\_facet
  - Locales, 180
  - std::locale, 2414
  - std::locale::id, 2419
- uses\_allocator< typename, typename >, 2891
- Utilities, 64
  - \_\_addressof, 67
  - addressof, 67
  - declval, 67
  - forward, 68
  - make\_pair, 68
  - move, 68
  - move\_if\_noexcept, 69
  - operator<, 69
  - operator<=, 69
  - operator>, 70
  - operator>=, 70
  - operator==, 69
  - piecewise\_construct, 71
  - swap, 70
  - tie, 70
  - tuple\_cat, 71
- utility, 3291
- valarray, 3292
  - Numeric Arrays, 84, 85
  - std::valarray, 2863
- valarray\_after.h, 3298
- valarray\_array.h, 3316
- valarray\_array.tcc, 3324
- valarray\_before.h, 3325
- valid\_prefix
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer, 1114
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter, 1117
- value
  - Regular Expressions, 210
- value\_comp
  - std::map, 2446
  - std::multimap, 2519
  - std::multiset, 2534
  - std::set, 2692
- value\_compare
  - std::set, 2681
- value\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator\_, 1014
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator\_, 1020
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_, 1030
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_, 1034
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_, 1075
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 1079
  - const\_iterator\_, 1267
  - iterator\_, 1273
  - point\_const\_iterator\_, 1277
  - point\_iterator\_, 1280
  - std::allocator\_traits, 1497
  - std::back\_insert\_iterator, 1541
  - std::complex, 2098
  - std::front\_insert\_iterator, 2243
  - std::insert\_iterator, 2300
  - std::istream\_iterator, 2375
  - std::istreambuf\_iterator, 2378
  - std::iterator, 2381
  - std::ostream\_iterator, 2608
  - std::ostreambuf\_iterator, 2611
  - std::raw\_storage\_iterator, 2651
  - std::reverse\_iterator, 2670
  - std::set, 2681
  - std::unordered\_map, 2789
  - std::unordered\_multimap, 2808
  - std::unordered\_multiset, 2826
  - std::unordered\_set, 2844
- vector, 3325, 3326
  - std::\_\_debug::vector, 1361
  - std::vector, 2867d
- vector.tcc, 3327
- void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits, 630
  - std::allocator\_traits, 1497
- vstring.h, 3328
- vstring.tcc, 3330
- vstring\_fwd.h, 3331
- vstring\_util.h, 3332
- wcerr
  - std, 565
- wcin
  - std, 565
- wclog
  - std, 565
- wcout
  - std, 565
- wcregex\_token\_iterator

- Regular Expressions, [187](#)
- wcsub\_match
  - Regular Expressions, [187](#)
- wfilebuf
  - I/O, [44](#)
- wfstream
  - I/O, [44](#)
- what
  - \_\_gnu\_cxx::forced\_error, [734](#)
  - \_\_gnu\_cxx::recursive\_init\_error, [753](#)
  - \_\_gnu\_pbds::container\_error, [1003](#)
  - \_\_gnu\_pbds::insert\_error, [1187](#)
  - \_\_gnu\_pbds::join\_error, [1188](#)
  - \_\_gnu\_pbds::resize\_error, [1205](#)
  - std::bad\_alloc, [1543](#)
  - std::bad\_cast, [1544](#)
  - std::bad\_exception, [1545](#)
  - std::bad\_function\_call, [1546](#)
  - std::bad\_typeid, [1547](#)
  - std::bad\_weak\_ptr, [1548](#)
  - std::domain\_error, [2204](#)
  - std::exception, [2211](#)
  - std::future\_error, [2251](#)
  - std::invalid\_argument, [2304](#)
  - std::ios\_base::failure, [2319](#)
  - std::length\_error, [2383](#)
  - std::logic\_error, [2421](#)
  - std::out\_of\_range, [2613](#)
  - std::overflow\_error, [2614](#)
  - std::range\_error, [2646](#)
  - std::regex\_error, [2654](#)
  - std::runtime\_error, [2674](#)
  - std::system\_error, [2716](#)
  - std::underflow\_error, [2774](#)
- widen
  - std::\_\_ctype\_abstract\_base, [1301](#), [1302](#)
  - std::basic\_fstream, [1611](#)
  - std::basic\_ifstream, [1651](#)
  - std::basic\_ios, [1675](#)
  - std::basic\_iostream, [1720](#)
  - std::basic\_istream, [1759](#)
  - std::basic\_istreamstream, [1800](#)
  - std::basic\_ofstream, [1833](#)
  - std::basic\_ostream, [1864](#)
  - std::basic\_ostreamstream, [1897](#)
  - std::basic\_stringstream, [2033](#)
  - std::ctype, [2118](#), [2119](#)
  - std::ctype< char >, [2129](#), [2130](#)
  - std::ctype< wchar\_t >, [2142](#)
  - std::ctype\_byname, [2154](#), [2155](#)
  - std::ctype\_byname< char >, [2165](#)
- width
  - std::basic\_fstream, [1611](#)
  - std::basic\_ifstream, [1652](#)
  - std::basic\_ios, [1675](#), [1676](#)
  - std::basic\_iostream, [1721](#)
  - std::basic\_istream, [1760](#)
  - std::basic\_istreamstream, [1801](#)
  - std::basic\_ofstream, [1833](#), [1834](#)
  - std::basic\_ostream, [1864](#), [1865](#)
  - std::basic\_ostreamstream, [1898](#)
  - std::basic\_stringstream, [2033](#)
  - std::ios\_base, [2313](#)
- wfstream
  - I/O, [44](#)
- wios
  - I/O, [45](#)
- wiostream
  - I/O, [45](#)
- wistream
  - I/O, [45](#)
- wistreamstream
  - I/O, [45](#)
- wofstream
  - I/O, [45](#)
- workstealing.h, [3333](#)
- wostream
  - I/O, [45](#)
- wostringstream
  - I/O, [45](#)
- wregex
  - Regular Expressions, [187](#)
- write
  - std::basic\_fstream, [1612](#)
  - std::basic\_iostream, [1722](#)
  - std::basic\_ofstream, [1834](#)
  - std::basic\_ostream, [1865](#)
  - std::basic\_ostreamstream, [1898](#)
  - std::basic\_stringstream, [2034](#)
- ws
  - std, [564](#)
- wsregex\_token\_iterator
  - Regular Expressions, [188](#)
- wssub\_match
  - Regular Expressions, [188](#)
- wstreambuf
  - I/O, [45](#)
- wstreampos
  - std, [497](#)
- wstring
  - Strings, [237](#)
- wstringbuf
  - I/O, [45](#)
- wstringstream
  - I/O, [45](#)
- xalloc
  - std::basic\_fstream, [1612](#)

- std::basic\_ifstream, [1652](#)
- std::basic\_ios, [1676](#)
- std::basic\_iostream, [1722](#)
- std::basic\_istream, [1760](#)
- std::basic\_istreamstream, [1801](#)
- std::basic\_ofstream, [1834](#)
- std::basic\_ostream, [1865](#)
- std::basic\_ostreamstream, [1899](#)
- std::basic\_stringstream, [2034](#)
- std::ios\_base, [2313](#)

#### xsggetn

- \_\_gnu\_cxx::enc\_filebuf, [728](#)
- \_\_gnu\_cxx::stdio\_filebuf, [780](#)
- \_\_gnu\_cxx::stdio\_sync\_filebuf, [798](#)
- std::basic\_filebuf, [1565](#)
- std::basic\_streambuf, [1928](#)
- std::basic\_stringbuf, [1989](#)

#### xspn

- \_\_gnu\_cxx::enc\_filebuf, [728](#)
- \_\_gnu\_cxx::stdio\_filebuf, [780](#)
- \_\_gnu\_cxx::stdio\_sync\_filebuf, [798](#)
- std::basic\_filebuf, [1566](#)
- std::basic\_streambuf, [1928](#)
- std::basic\_stringbuf, [1990](#)

#### yield

- std::this\_thread, [619](#)